Machine Learning: Science and Technology

Recipes for when physics fails: recovering robust learning of physics informed neural networks

To cite this article: Chandrajit Bajaj *et al* 2023 *Mach. Learn.: Sci. Technol.* **4** 015013

IOP Publishing Mach. Learn.: Sci. Technol. 4 (2023) 015013 https://doi.org/10.1088/2632-2153/acb416

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
26 July 2022

REVISED
25 November 2022

ACCEPTED FOR PUBLICATION
17 January 2023

PUBLISHED
6 February 2023

PAPER

Recipes for when physics fails: recovering robust learning of physics informed neural networks

Chandrajit Bajaj[1], **Luke McLennan**[1], **Timothy Andeen**[2] **and Avik Roy**[3]

[1] Department of Computer Science & Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, 78712, United States of America
[2] Department of Physics, The University of Texas at Austin, Austin, TX, 78712, United States of America
[3] Center for AI Innovation, National Center for Supercomputing Applications, University of Illinois at Urbana Champaign, Urbana, IL, 61801, United States of America

E-mail: avroy@illinois.edu

Keywords: Physics informed deep learning, Gaussian Processes, nonlinear PDE, Robust deep learning

Abstract

Physics-informed neural networks (PINNs) have been shown to be effective in solving partial differential equations by capturing the physics induced constraints as a part of the training loss function. This paper shows that a PINN can be sensitive to errors in training data and overfit itself in dynamically propagating these errors over the domain of the solution of the PDE. It also shows how physical regularizations based on continuity criteria and conservation laws fail to address this issue and rather introduce problems of their own causing the deep network to converge to a physics-obeying local minimum instead of the global minimum. We introduce Gaussian process (GP) based smoothing that recovers the performance of a PINN and promises a robust architecture against noise/errors in measurements. Additionally, we illustrate an inexpensive method of quantifying the evolution of uncertainty based on the variance estimation of GPs on boundary data. Robust PINN performance is also shown to be achievable by choice of sparse sets of inducing points based on sparsely induced GPs. We demonstrate the performance of our proposed methods and compare the results from existing benchmark models in literature for time-dependent Schrödinger and Burgers' equations.

1. Introduction

Neural networks (NNs) are finding ubiquitous applications in fundamental sciences. Their abilities to perform classification and regression over large and complicated datasets are making them extremely useful for a variety of purposes, including modeling molecular dynamics modeling, nonlinear dynamical system design and control, and other many body interactions [1–3]. In many cases, these important problems in physics and engineering are posed in terms of static and time dependent partial differential equations (PDEs). Analytical solutions to PDEs are scarce. Moreover, PDEs are difficult to solve and require computationally intensive, and highly pre-conditioned numerical linear solvers [4]. Popular discretization methods, finite difference method (FDM), and even finite element method (FEM) are used to obtain point wise or piece-wise linear estimates over a fine grid or meshed domains of interest [5]. Although NN-based approximations to differential equations have been explored for some time [6–8], interest in such approaches have been reinvigorated recently due to significant improvement in computational platforms that support fast forward and backward gradient propagation utilizing automatic differentiation [9]. Rapid progress has been seen in NN-assisted solutions of ordinary differential equations (ODEs) [10]. Novel architectures like NeuralODEs [11] have been proposed to harness the power of blackbox ODE solvers in conjunction with continuous-depth residual NNs (Resnets) using the method of adjoints [12], while models like ODE2VAEs use variational auto-encoder architectures to learn functions and derivatives via latent space embeddings similar to Cauchy boundary conditions [13].

Solving PDEs using NNs has also seen significant attention by the development of physics informed neural networks (PINNs) [14]. Trained PINNs have been shown to be effective in solving time dependent PDEs for a given set of Cauchy boundary conditions over a finite spatio-temporal domain. They exploit a deep NN's ability as universal function approximators [15, 16]. Many variants of the trainable PINN architecture have been explored to exploit structure, quality and speed of convergence, and dimensional scalability [17–23]. However, one of the least explored areas is the robustness of trainable PINNs for various noisy data scenarios and conditions. While most PINN architectures in literature assume perfectly known boundary data, in many practical applications, this data comes from regulated and calibrated measurement processes and is subject to uncertainties or errors pertaining to the limitations of the measurement system or the stochastic nature of the physical processes themselves. Most PINN architectures utilize a finite and small collection of training data on the domain boundary. Given the typical small size of this training data and NNs ability to capture arbitrary non-linearity, vanilla PINNs can often propagate these errors or uncertainties in an unstable fashion. Such unregulated error propagation can significantly limit the applicability of PINNs as industrial strength numerical approximators of PDEs.

In this paper, we extensively investigate the problem of error propagation in PINNs. In section 2 we introspect the architecture of a PINN, and how it responds when data on the initial timeslice is corrupted with noise. The notion of PINN robustness is thus tied to the PINNs ability to preserve solution features under such noisy perturbations. We analyze examples of learning time-varying non-linear Schrödinger, and the non-convective fluid flow Burgers' equations. We further show the impact of introducing regularization based on continuity criteria inspired from conservative PINN (cPINN) architectures [19, 22] through attempts to satisfy conservation laws. In section 3 we provide details of a Gaussian process smoothed PINN (GP-smoothed PINN), and its sparse variant and compare its performance against most typical and popular PINN architectures. We demonstrate how these twin GP-smoothed PINNs recover the intended solution and can outperform methods realizing continuity or conservation regularizers [22] as well as better possess the ability to control uncertainty propagation compared to uncertainty quantification methods proposed for example in [20]. We briefly examine the efficiency role of sparse inducing points (IPs) and also demonstrate the importance and choice of various kernels for achieving best model selection in GP training to prevent data driven under- or overfitting.

2. Robustness of PINNs

In this section, we review the PINN and cPINN architectures and explain with different examples how such models fail to capture the essence of robustness. We also investigate multiple physics-inspired regularization schemes and identify their limitations in addressing the issue of robustness for usual PINN architectures.

2.1. Review of PINN architectures

A PDE that determines spatio-temporal evolution of a set of scalar (real or complex) fields, collectively represented by $u(\vec{x})$, can be expressed as

$$\mathcal{N}[u(\vec{x}), f(\vec{x})] = 0 \tag{1}$$

where $\vec{x}$ defines a n dimensional spatial or saptio-temporal coordinate system, defined on the domain $\vec{x} \in \mathcal{D} \subset \mathbb{R}^n$. $\mathcal{N}$ represents a set of known, finite-order, differential operators and $f(\vec{x})$ is the source function, usually known as analytical expression in problems intended to solve a forward PDE problem. Equation (1) is subject to a set of boundary conditions,

$$\mathcal{B}[u(\vec{x} \in \partial D)] = 0. \tag{2}$$

The general idea of a PINN [14] is to obtain an approximation of the field $u(\vec{x}) \approx \tilde{u}(\vec{x})$ by a deep NN that can solve the system of equation (1), subject to 2.

$$\tilde{u}(\vec{x}) = \mathbf{NN}_\theta(\vec{x}; \mathcal{U}_B, \mathcal{U}_C, \mathcal{U}_D) \tag{3}$$

where,

- $\mathcal{U}_B = \{(\vec{x}_i^b, \mathcal{B}[u(\vec{x}_i^b)])_{i=1}^{N_b}\}$ represents a set of samples on the domain boundary $\partial\mathcal{D}$,
- $\mathcal{U}_C = \{(\vec{x}_i^c, f(\vec{x}_i^c))_{i=1}^{N_c}\}$ is a set of measurements that enforces the PDE physics of equation (1) on the neural net (equation (3)), and
- $\mathcal{U}_D = \{(\vec{x}_i^d, u(\vec{x}_i^d))_{i=1}^{N_d}\}$ is a set of direct measurements. Although, $\mathcal{U}_D$ is not necessary for training a plain PINN as in [14], they are often used for training networks for targeted simulation.

The parameters of the deep network (θ) in equation (27) are obtained by minimization of the loss function,

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{\text{PINN}} \tag{4}$$

where the loss function can be decomposed as,

$$\mathcal{L}_{\text{PINN}} = \alpha_{\text{BC}}\mathcal{L}_{\text{BC}} + \alpha_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \alpha_D\mathcal{L}_D. \tag{5}$$

Here, $\mathcal{L}_{\text{BC}}$ is the MSE loss calculated over $\mathcal{U}_B$, enforcing the NN to approximate the boundary condition,

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_b}\sum_i \left|\mathcal{B}[\tilde{u}(\vec{x}_i^b)]\right|^2 \tag{6}$$

$\mathcal{L}_{\text{PDE}}$ is the MSE loss calculated over $\mathcal{U}_C$, enforcing the physics on the NN,

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_c}\sum_i |\mathcal{N}[\tilde{u}(\vec{x}_i^c), f(\vec{x}_i^c)]|^2 \tag{7}$$

and finally, $\mathcal{L}_D$, if employed, determines the loss with respect to observation,

$$\mathcal{L}_D = \frac{1}{N_d}\sum_i \left|\tilde{u}(\vec{x}_i^d) - u(\vec{x}_i^d)\right|^2. \tag{8}$$

The $\alpha_{()}$ parameters in equation (5) are penalty parameters that determine the relative strength of the regularizing terms in the loss function. Authors in [14] assign $\alpha_{()} = 1$ identically, but alternate choices have been explored in other works [22, 24].

2.1.1. cPINNs

Variations of PINN architectures have been explored in a number of recent works. For example, authors in [21] explore a multi-staged PDE solver for a long range solution and [19] introduces adaptive, hyperparameterized activation functions that would accelerate the convergence of such networks.

Although the training of PINNs does not strictly require a discretized grid of evaluation points as often required by traditional numerical techniques like FDM and FEMs, they can certainly benefit from such grid structures by parallelizing the training of PINNs over a collection of subdomains. Hence, the domain of integration and its boundary are divided into subdomains, i.e.

$$\mathcal{D} = \bigcup_{i=1}^{K} d_i, \qquad \partial\mathcal{D} = \bigcup_{i=1}^{K} \partial d_i, \qquad \tilde{u} = \bigcup_{i=1}^{K} \tilde{u}_i. \tag{9}$$

An additional benefit to such parallelized structure is to include flux continuity at subdomain boundaries in the loss function, providing additional safeguard against error propagation. In other words, the loss function in equation (5) is modified as

$$\mathcal{L}_{\text{cPINN}} = \sum_{j=1}^{d} \mathcal{L}^j_{\text{PINN}} + \alpha_I\mathcal{L}^j_I \tag{10}$$

where $\mathcal{L}^j_{\text{PINN}}$ is the PINN loss for the jth subdomain as defined in equation (5). The other term, $\mathcal{L}^j_I$ acts as a regularizer that enforces functional and flux continuity at the interface of jth subdomain interface

$$\mathcal{L}^j_I = \frac{1}{N_{Ij}}\sum_{i=1}^{N_{Ij}} \left(\left|\tilde{u}_j(\vec{x}_i^j) - \tilde{u}_{j+1}(\vec{x}_i^j)\right|^2 + \left|\nabla\tilde{u}_j(\vec{x}_i^j)\cdot\mathbf{n}_\mathbf{i}^\mathbf{j} - \nabla\tilde{u}_{j+1}(\vec{x}_i^j)\cdot\mathbf{n}_\mathbf{i}^{\mathbf{j}+1}\right|^2 \right). \tag{11}$$

here, $\{\vec{x}_i^j\}$ is a collection of points on the interface of jth and $j+1$th subdomains and $\mathbf{n}_\mathbf{i}^\mathbf{j}$ is the unit vector normal to the interface of jth subdomain at the location of $\vec{x}_i^j$.

Figure 1 shows a schematic representation of the different spatio-temporal regions that contribute to evaluating the loss function in equation (5) along with a generic architecture for such models.

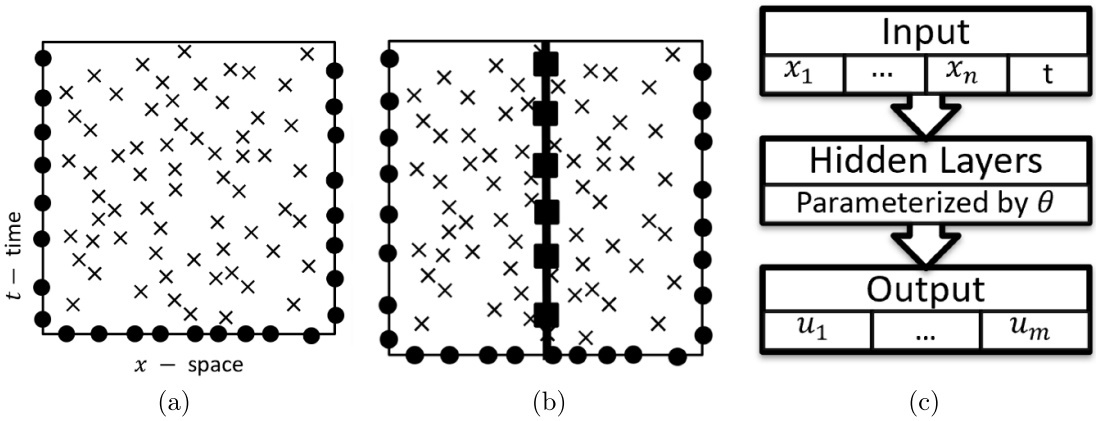


Figure 1. A visualization of the domains of (a) a PINN and (b) a cPINN with boundary points (•), collocation points (×), and interface points (■) for a spatio-temporal domain with one spatial dimension. Figure (c) shows the model architecture diagram of a PINN for a generic $n+1$ dimensional spatio-temporal domain solving for a set of coupled fields $u_1, \ldots, u_m$.

2.2. Error propagation through PINNs

An NN can approximate non-linear functions with increasing degrees of accuracy. It is typically expected in the case of a PINN that $N_b \ll N_c$ i.e. the size of the training data on the domain boundary is usually much smaller than the size of the physics-enforcing collocation points. This unavoidable feature of PINNs make them susceptible to overfitting on the boundary and eventually propagate those errors across the domain. In the following subsections, we investigate the physical nature of error propagation through PINNs and the impact of different regularizations on training the PINN architecture. We will use two popular examples that have been widely used in the literature to illustrate this issue of error propagation.

2.2.1. Nonlinear Schrödinger equation

We consider the example considered in [14] of a nonlinear Schrödinger PDE which describes the spatio-temporal evolution of a 1D complex field $h(x,t) = u(x,t) + iv(x,t)$ as

$$i\frac{\partial h}{\partial t} + \frac{1}{2}\frac{\partial^2 h}{\partial x^2} + |h|^2 h = 0 \tag{12}$$

which can also be interpreted as a set of coupled PDEs given as

$$\begin{aligned} -\frac{\partial v}{\partial t} + \frac{1}{2}\frac{\partial^2 u}{\partial x^2} + (u^2+v^2)u = 0 \\ \frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial^2 v}{\partial x^2} + (u^2+v^2)v = 0. \end{aligned} \tag{13}$$

The domain boundary is defined as $(x,t) \in [-5,5] \times [0, \frac{\pi}{2}]$. The boundary conditions can be classified as

(a) Initial condition; the known value of the field on the initial time slice given as a collection of measurements $\{(x_j, h(x_j,0)_{j=0}^{N_{b,t}}\}$. Unbeknownst to the NN, these measurements are taken from the analytical solutions with possible sources of additive corruption-

$$\begin{aligned} u(x_j,0) &= 2\text{sech}(x_j) + \Theta_u \epsilon_i^u \\ v(x_j,0) &= \Theta_v \epsilon_i^v \end{aligned} \tag{14}$$

where ϵ^u, ϵ^v are randomly chosen from a normal distribution with $\mathcal{N}(0,\sigma^2)$. The parameters Θ_u and Θ_v represent acceptance of errors which are set to 0 (1) for error-free (error-inclusive) initial conditions.

(b) Periodic boundary condition on spatial slices enforced on a discretized spatial boundary. A total of $N_{b,s}$ points are chosen to enforce the following spatial boundary conditions

$$\begin{aligned} h(+5,t) &= h(-5,t) \\ \frac{\partial h}{\partial x}(+5,t) &= \frac{\partial h}{\partial x}(-5,t). \end{aligned} \tag{15}$$

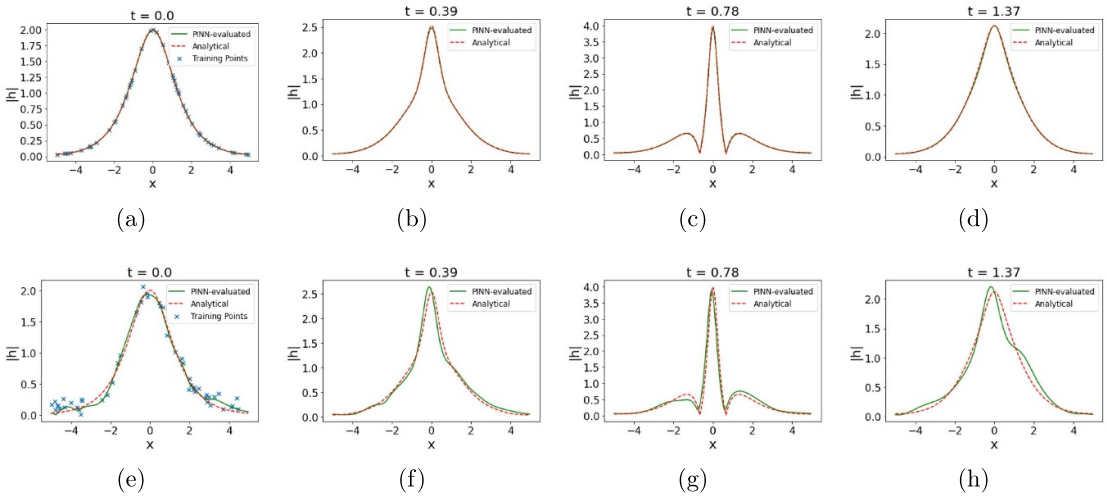


Figure 2. The PINN-evaluated solution for $|h(x,t)|$ from the Schrödinger equation at different timeslices, $t = 0, 0.39, 0.78, 1.37$, from left to right for error-free boundary data (top) and corrupted boundary data with (bottom). The additive errors on the boundary data are taken independently from samples of zero mean Gaussian distribution with $\sigma = 0.1$. The points marked with the blue cross (x) pointer in the leftmost set of plots indicate the samples on the initial timeslice used to train the PINN.

The loss function is constructed according to equation (5) with $\alpha_{()} = 1.0$. To set the benchmark for the performance of a PINN for this problem, we solved for the PDE with error-free boundary data ($\Theta_u = \Theta_v = 0$). We train a fully connected MLP with six hidden layers and 70 nodes per layer, with two inputs corresponding to the space and time coordinate and two outputs corresponding to the real and imaginary parts of the complex field. The MLP was trained for $N_b = 100$ points on the domain boundary, 50 of which were taken from uniformly sampling the space coordinate (x) on the initial timeslice to impose the initial condition in equation (14), and the 50 points were taken on from a uniform distribution on the time coordinate to impose the periodic boundary condition in equation (15). A fine grid of $N_c = 20000$ collocation points was chosen to impose the physics of the PDE.

Figure 2 shows the evolution of the complex field in the Schrödinger equation as evaluated by a vanilla PINN at four different timeslices, taken at $t = 0, 0.39, 0.78, 1.37$. The performance of the PINN for error-free data on the initial timeslice is shown in figures 2(a)–(d). We use the same architecture to repeat the exercise while training on corrupted data on initial timeslice by letting $\Theta_u = \Theta_v = 1$ and additive errors generated by drawing samples from zero-mean Gaussian distribution with $\sigma = 0.1$. The performance of the PINN in evaluating the complex field magnitude at the same time instances is shown in figures 2(e)–(h). The effect of introducing corrupted data on initial timeslice becomes evident when comparing figures 2(a)–(d) with figures 2(e)–(h). The PINN tends to overfit on the initial timeslice and eventually propagates these errors on the following timeslices.

2.2.2. Burgers' equation

We consider the Burgers' equation in one spatial dimension with Dirchlet boundary conditions as a second example. Widely used in fluid dynamics and nonlinear acoustics, this nonlinear PDE has been widely studied as a benchmark example in the PINN literature [14, 20]. The PDE and the boundary conditions for 1D Burgers' equation are given as :

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{16}$$

$$u(-1,t) = u(1,t) = 0 \tag{17}$$

$$u(x,0) = -\sin(\pi x) + \Theta_u \epsilon^u \tag{18}$$

where the domain boundary is given as $(x,t) \in [-1,1] \times [0,1]$. To set the benchmark for this problem, we solve equation (16) subject to boundary conditions in equations (17) and (18) with $\nu = \frac{0.01}{\pi}$ using a PINN without noise ($\Theta_u = 0$) in the initial data. We used an MLP with 4 hidden layers, each with 40 nodes. We trained with $N_c = 10000$ collocation points to enforce the physics (equation (16)) and $N_b = 150$ points on the boundary, 50 points for enforcing the initial condition in equation (18) and 50 points on each of the

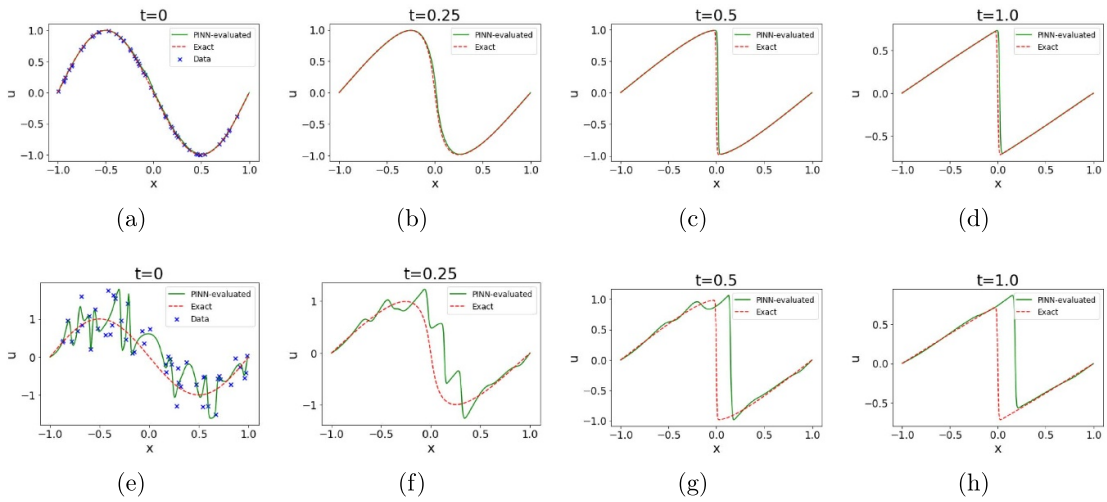


Figure 3. The PINN-evaluated solution for $u(x,t)$ from the Burgers' equation at different timeslices, $t = 0, 0.25, 0.50, 1.00$, from left to right for error-free boundary data (top) and corrupted boundary data with (bottom). The additive errors on the boundary data are taken from independently samples of zero mean Gaussian distribution with $\sigma = 0.5$. The points marked with the blue cross (x) pointer in the leftmost set of plots indicate the samples on the initial timeslice used to train the PINN.

spatial boundaries at $x = -1, 1$ to enforce each of the Dirichlet conditions in equation (17). The loss function is constructed according to equation (5) with $\alpha_{()} = 1.0$. Figures 3(a)–(d) shows the evolution of the field $u(x,t)$ in the Burgers' equation as evaluated by a vanilla PINN at four different timeslices, taken at $t = 0, 0.25, 0.50, 1.0$.

Next, we repeat the exercise for Burgers' equation by introducing additive corruption with $\Theta_u = 1$ and $\sigma = 0.5$. Figures 3(e)–(h) show the corresponding line shapes of the Burgers' field for the aforementioned timestamps. Similar to what was found for the Schrödinger equation, the vanilla PINN architecture fails to auto-correct for the corruption in initial data and ends up overfitting on the initial timeslice and eventually propagates these errors to later timeslices.

2.3. Regularization of PINNs

Both figures 2 and 3 show a PINN's inherent inability to self-correct when trained with error-corrupted data on the initial timeslice. The PINN rather learns to overfit on the initial timeslice and eventually propagates the initial errors to later timeslices. This propagation of error to later timeslices is a direct consequence of having no regularization in the loss term in equation (5) to constrain overfitting at the domain boundary.

In principle, this is not very different from overfitting in classic regression problems with a high degree polynomial or introduction of bias in an un-regularized regression by outliers. This naively indicates that additional regularization might be useful to limit the propagation of errors. However, our investigations indicates that some of the most physically intuitive choices for regularizing constraints have little impact on error propagation and boundary overfitting.

We consider two unique choices of regularizers. First, inspired by the cPINN architecture, we impose the constraint of functional and flux continuity at arbitrary spatio-temporal boundaries to constrain the PDE solution. Second, we explore the possibility of using physical conservation laws as additional sources of regularization. In the following subsections, we explore the impact of using such regularization schemes in training PINNs with corrupted boundary data.

2.3.1. Functional and flux continuity at subdomain interfaces

The cPINN architecture inspires a useful regularization that imposes continuity of the field and its flux across domain boundaries. From a physics standpoint, these regularizations can be thought of as additional conservation laws that ensures continuity and differentiability of a field across subdomains. In this subsection, we explore the impact of including this term in the training loss function in equation (10) in controlling propagation of uncorrelated errors at sampling points on the initial timeslice.

One immediate concern is that the convergence of the cPINN is susceptible to exact choices of how many subdomains are chosen and where those boundaries are located. Based on the location of the subdomain boundaries, the cPINN's capacity to converge to the global minimum of the training loss function, can be significantly impacted. To illustrate this, we compare the performance of cPINNs in solving the Schrödinger

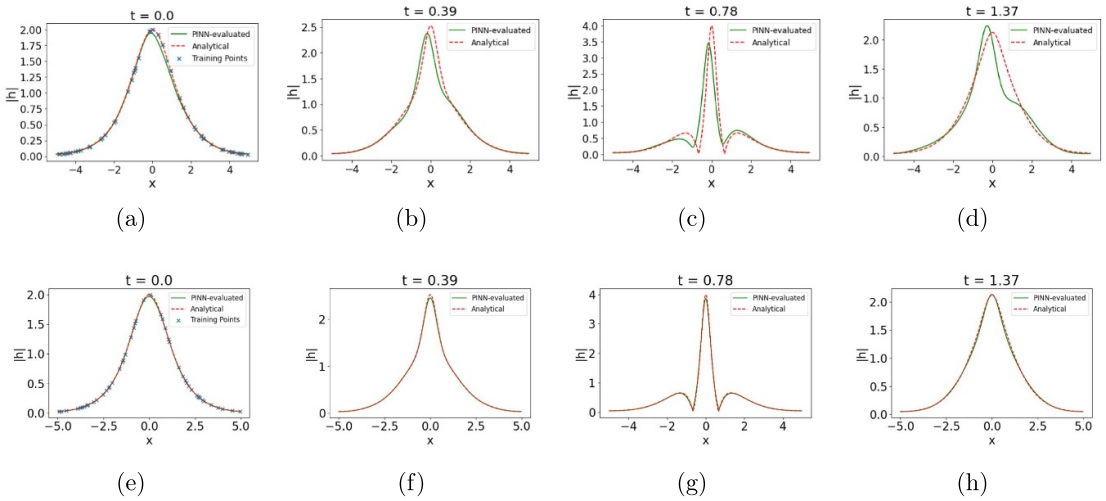


Figure 4. The cPINN-evaluated $|h(x,t)|$ for two (top row) and three (bottom row) equal subdomains at different timeslices, $t = 0, 0.39, 0.78, 1.37$, from left to right when no error is introduced on initial time-slice. The points marked with the blue cross (x) pointer in the leftmost set of plots indicate the samples on the initial timeslice used to train the cPINN.

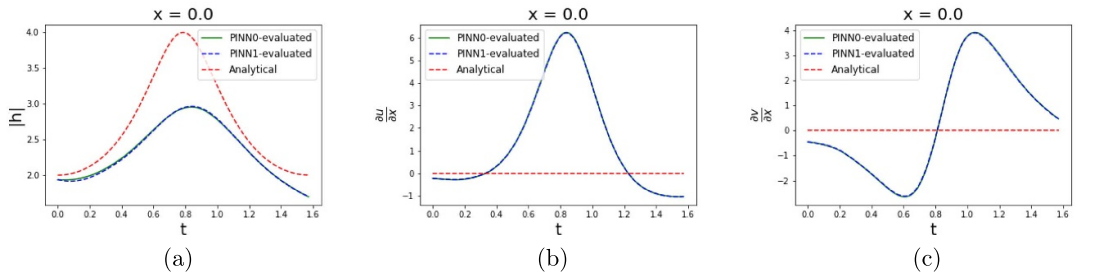


Figure 5. The cPINN evaluated lineshape for $|h(x,t)|$ (left), $\frac{\partial u}{\partial x}$ (middle), and $\frac{\partial v}{\partial x}$ (right) at the subdomain interface ($x = 0$) for a two subdomain cPINN as a function of t. `PINN0` (`PINN1`) refers to the PINN trained to solve the PDE on a spatial boundary of $-5 \leqslant x \leqslant 0$ $(0 \leqslant x \leqslant 5)$.

equation with two and three equal spatial subdomains trained on error-free boundary data. The results are shown in figure 4. Evidently, a three subdomain cPINN better recovers the analytical solution. However, the failure of a two subdomain cPINN, to capture the solution of a PDE even for error-free boundary data is intriguing. This observation yields a deeper insight to the impact of adding additional regularizers with the PINN loss function in equation (5). As can be seen in figure 4(a), the two subdomain cPINN moderately deviates from the analytical solution on the initial timeslice at the expense of converging to the local minima introduced by the inclusion of the interface loss. Figure 5 shows how the PINNs trained on different subdomains converge to identical functional and flux values at the subdomain boundary but eventually experiences large deviations from the analytical solution which requires $\frac{\partial u}{\partial x}(x,0) = \frac{\partial v}{\partial x}(x,0) = 0$. It is apparent that choice of subdomain boundary at $x = 0$ plays an important role in causing such deviating solutions. As the real and imaginary fields reach local extrema at $x = 0$ for all values of t, even small deviations in estimating the local gradients at the interface can create a cascading effect in the evaluation of the complex field across subdomain boundaries as we can see in figures 5.

The result of such instabilities in convergence, is evidential consequence of such cPINN-inspired regularizations and the architecture's failure to be robust to such noisy perturbations. As also shown in figure 6, regularization of functional and flux continuity at subdomain boundaries does not provide the necessary safeguard to ensure robustness.

When we repeat the same set of exercises for the Burgers' equation, we see very similar results, as shown in figures 7. Similar to what we have seen for the Schrödinger equation, placing the subdomain boundaries at functionally critical points can destabilize and deteriorate the quality of the solution learned by the deep network. It can be seen in figures 7(a)–(d) from the deviation of the Burgers' field's predicted behavior at later timeslices even without any error introduced on the initial timeslice when two subdomains are considered with an interface at $x = 0$. However, the function is almost identically recovered when the data is

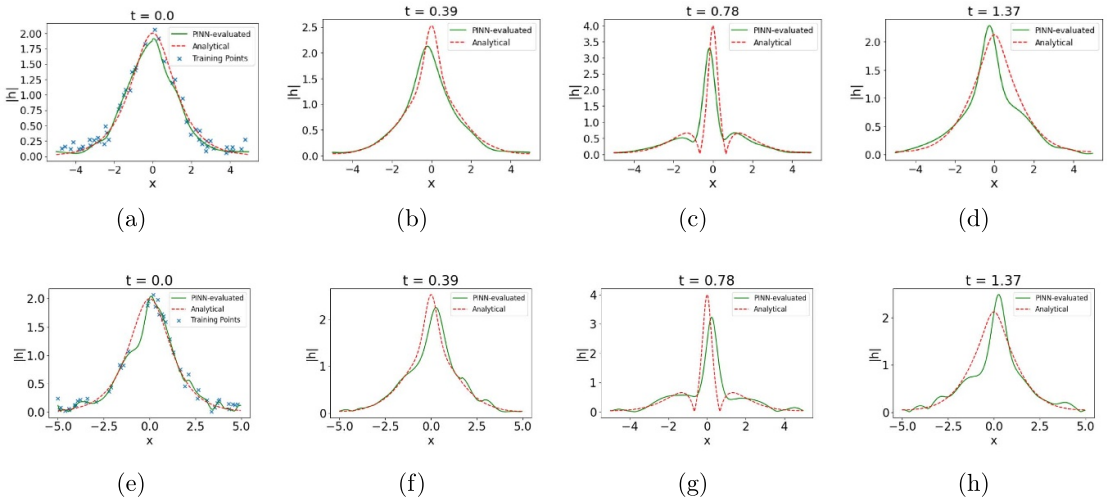


Figure 6. The cPINN-evaluated $|h(x,t)|$ for two (top row) and three (bottom row) subdomains at different timeslices, $t = 0, 0.39, 0.78, 1.37$, from left to right when additive Gaussian errors with $\sigma = 0.1$ is introduced on initial time-slice. The points marked with the blue cross (x) pointer in the leftmost set of plots indicate the samples on the initial timeslice used to train the cPINN.

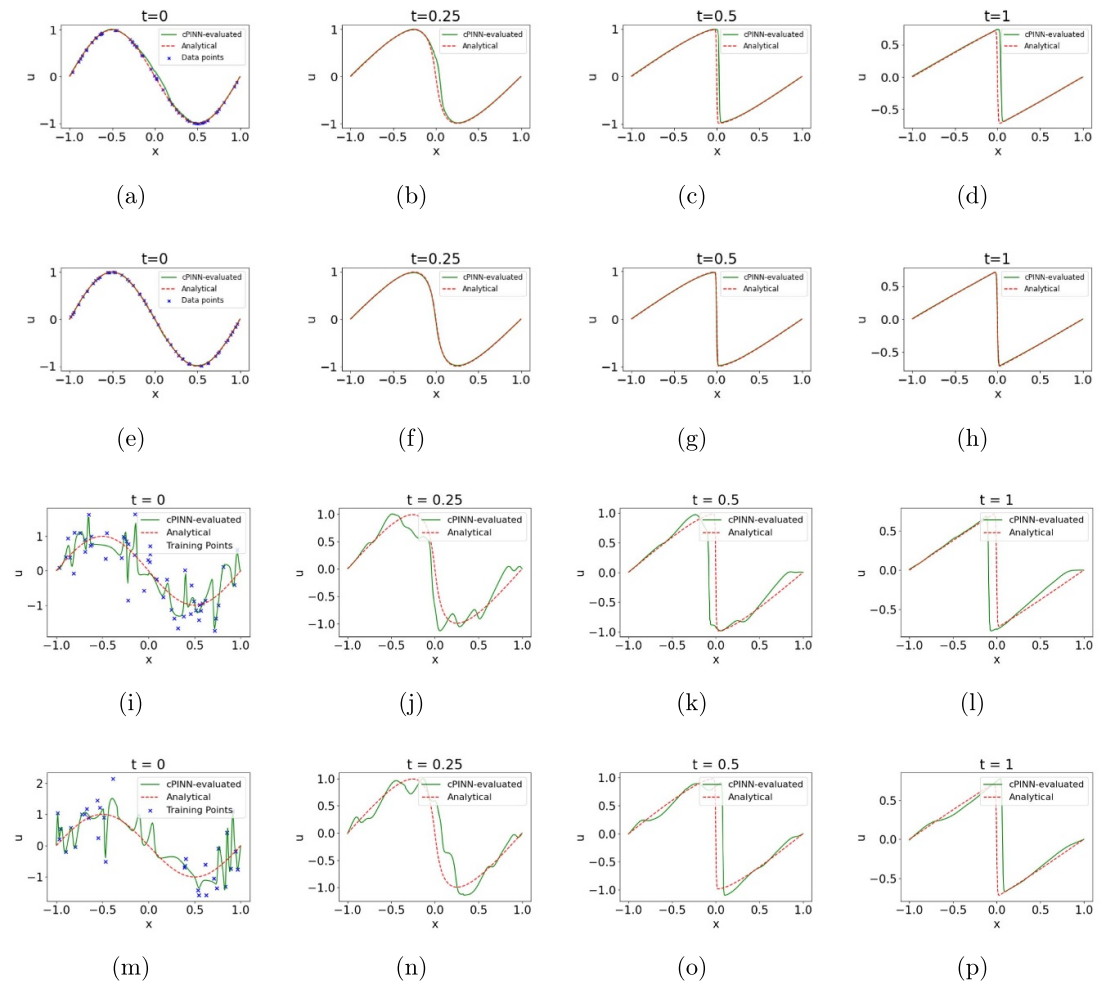


Figure 7. cPINN-evaluated solution to Burgers' equation with (a)–(d) two subdomains with no error, (e)–(h) three subdomains with no error, (i)–(l) two subdomains with error, and (m)–(p) three subdomains with error. The error on individual datapoints, whenever applied, have been taken from additive zero-mean Gaussian distribution with $\sigma = 0.5$.

perfect on the initial timelice with three subdomains as shown in figures 7(e)–(h). This tendency of a PINN-like architecture to converge to a local minima instead of the global minima almost infallibly deteriorates the quality of convergence when error is introduced on the initial timeslice, and thus the solution departs significantly from its ideal behavior. This is apparent in both the two subdomain (figures 7(i)–(l)) and the three subdomain cPINNs (figures 7(m)–(p)).

2.3.2. Conservation law constraints

Physical laws are often subject to a number of conservation laws. While in many cases these conservation laws emerge as direct consequences of the mathematical structure of the PDE, explicitly enforcing such conservation laws will back-propagate additional constraining gradients for the NN hyperparameters. We can explicitly include these conservation laws into the loss function.

For example, one of the major consequences of non-linear Schrödinger equation is global conservation of the squared absolute value of the complex Schrödinger field $|h(x,t)|^2$

$$\int |h(x,t)|^2 \mathrm{d}x = \int (u(x,t)^2 + v(x,t)^2)\mathrm{d}x = C \tag{19}$$

where C is a constant. A number of other conserved quantities follow for the 1D nonlinear Schrödinger equation we are considering [25–27], which include:

$$\int \left(u\frac{\partial v}{\partial x} + v\frac{\partial u}{\partial x} \right) \mathrm{d}x \tag{20}$$

$$\int \left(\left| \frac{\partial h}{\partial x} \right|^2 - |h|^4 \right) \mathrm{d}x. \tag{21}$$

We can constrain the solution by explicitly including these conservation laws as regularizers in the loss function. For instance, the probability conservation law in equation (19), along with the requirement of probability confinement within the spatially bounded region for the domain of equation (12) requires that

$$0 = \frac{d}{\mathrm{d}t}\int_a^b |h|^2 \mathrm{d}x = \frac{d}{\mathrm{d}t}\int_a^b u^2 + v^2 \mathrm{d}x = 2\int_a^b uu_t + vv_t \mathrm{d}x \tag{22}$$

where (a, b) is the spatial domain. We can approximate this integral using time-sliced collocation points:

$$\int_a^b uu_t + vv_t \mathrm{d}x \approx \frac{b-a}{n_{c,t}} \sum_{i=1}^{n_{c,t}} u(x_i,t)u_t(x_i,t) + v(x_i,t)v_t(x_i,t) \tag{23}$$

where $n_{c,t}$ represents the number of collocation points chosen over the spatial subdomain at timeslice t. We can define the conservation loss to be

$$\mathcal{L}_C = \frac{1}{N_t}\sum_{i=0}^{N_t} \frac{1}{n_{c,t}} \sum_{j=0}^{n_{c,t}} \left(u\left(x_j,t_i\right) u_t\left(x_j,t_i\right) + v\left(x_j,t_i\right) v_t\left(x_j,t_i\right) \right)^2. \tag{24}$$

We train a PINN with this conservative constraint on data appended with the loss function in equation (5). As seen in figure 8, the constrained PINN better represents the conservation laws, with the overall range for dispersion of cumulative probability effectively reduced with the inclusion of conservative constraints in the PINN loss function. However, the observed evolution of the field in figure 9, obtained by applying this constraint does not result in superior accuracy. Errors are still propagated from the initial timeslice throughout the spatio-temporal domain.

As a second example of the implication of conservation laws as regularizers, we take the example of Cole-Hopf transformation [28, 29] for the Burgers' equation. Based on proper mathematical wisdom developing analytical solutions to PDEs, methods like the Cole-Hopf transform have been found useful to convert one family of PDEs into another whose analytical solution is known and rather simple to compute. The Cole-Hopf transformation converts the nonlinear Burgers' equation to a linear Heat equation. This transformation is defined by making the following change of variables:

$$u(x,t) = -2\nu\frac{\partial v}{\partial x}(x,t). \tag{25}$$

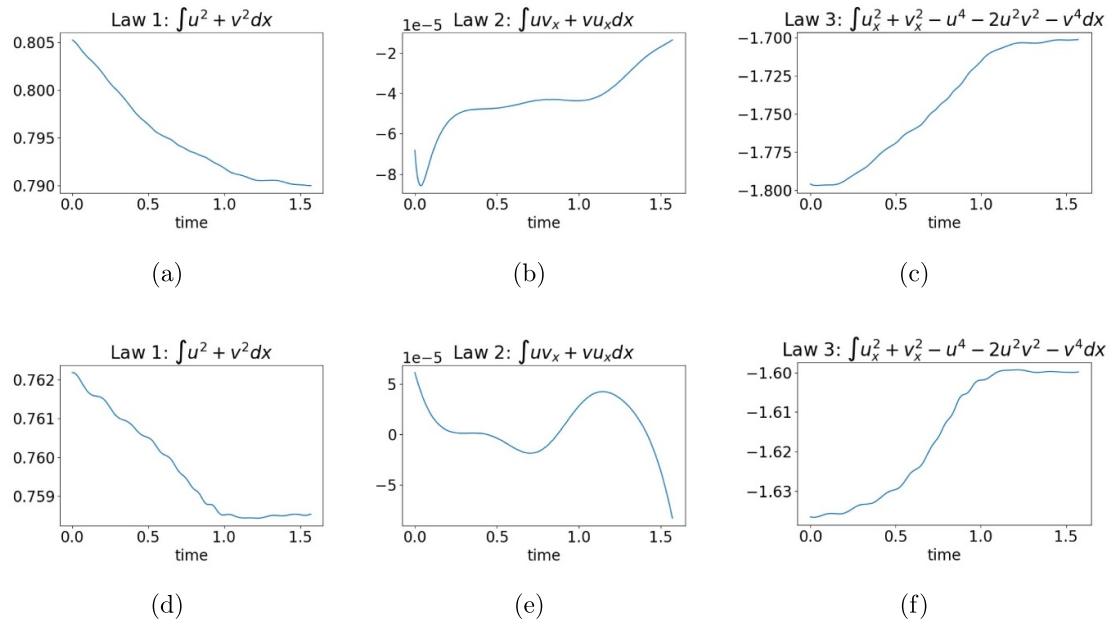


Figure 8. Plots of three of the conservation laws (equations (19)–(21)) of the nonlinear Schrödinger equation for (a)–(c) a vanilla PINN and (d)–(f) a PINN constrained by the first conservation law. The PINNs were trained with initial data corrupted with $\sigma = 0.1$ Gaussian noise. The constrained PINN has better performance of conservation laws 1 and 3, while both PINNs satisfy conservation law 2 nearly exactly.

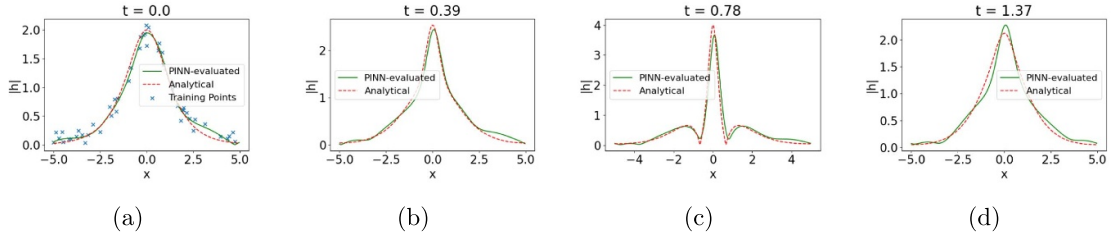


Figure 9. The PINN-evaluated $|h(x,t)|$ at different timeslices, $t = 0, 0.39, 0.78, 1.37$, from left to right. In this case the PINN is constained by the Schrodinger equations first conservation law: $\frac{d}{dt}\int |h|^2 dx = 0$. The training data on the initial timeslice is subject to measurement errors, modeled by a Gaussian random variable with zero mean and a standard deviation of $\sigma = 0.1$. The points marked with the blue cross (x) pointer in the leftmost set of plots indicate the samples on the initial timeslice used to train the PINN.

The transformed field $v(x,t)$ satisfies the heat equation $\partial_t v = \nu \partial_{xx} v$. The viscous case of the Burgers' equation is for $\nu > 0$ causing a non-linear dissipative shock for small values of ν. The inviscid case yields the equation having a non-linear hyperbolic conservation law. For our purpose, the conservation law for the modified field can be viewed as a regularizer to the conventional PINN loss function in equation (5).

$$\mathcal{L}_{\text{CH}} = \sum_{i,j} \left(v_t(x_i, t_j) - \nu v_{xx}(x_i, t_j) \right)^2 . \tag{26}$$

In addition to using the Cole-Hopf loss term as a regularizer for a vanilla PINN architecture, we additionally consider including the continuity criteria for equally split two and three subdomains. Figures 10(a)–(d) shows the time evolution of the Burgers' field $u(x,t)$ when the PINN is trained with the loss function including the Cole-Hopf term in equation (26). The functional approximation obtained from the PINN is much smoother compared to what we have observed in our previous exploration of regularized evaluation of the solution to the Burgers' equation (figures 3 and 7). It can be directly traced back to the fact that the Cole-Hopf transformed field is indeed an anti-derivative of the Burgers' field and in the neural architecture when implemented as a discrete integration acts as a smoothing operation in somewhat canceling out the effect of the error. The smoothing operation however eventually leads to underfitting and the evolution of the field at later timeslices is affected in a similar fashion.

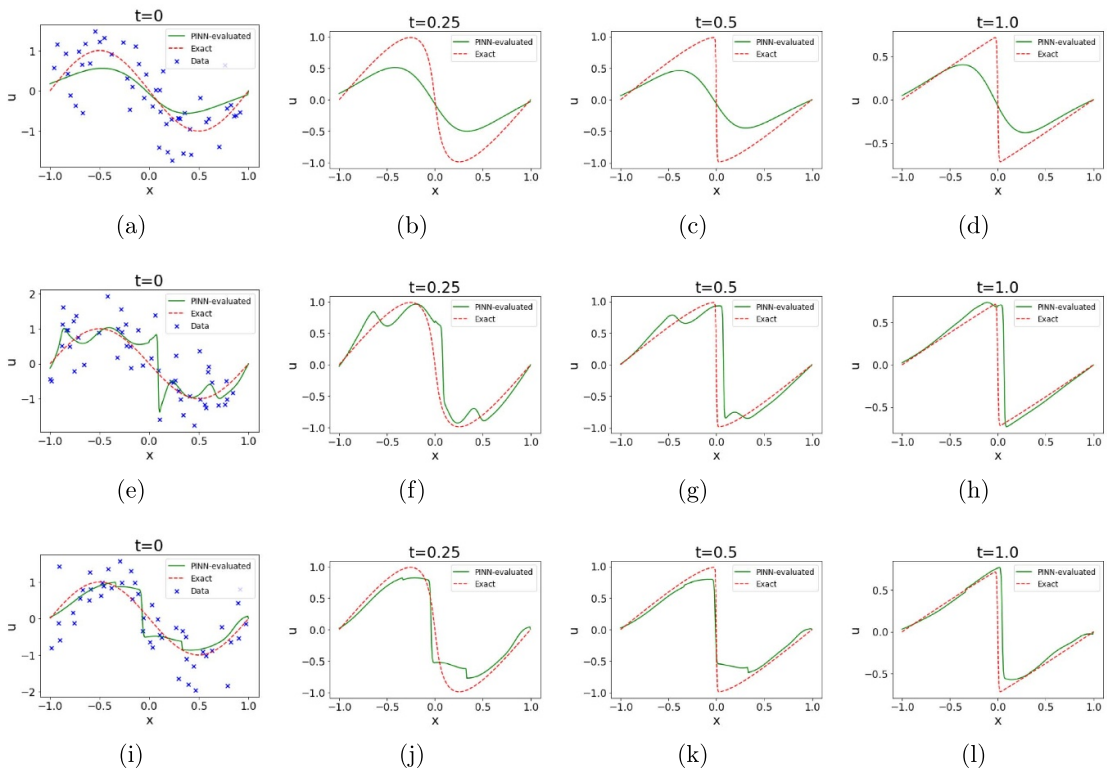


Figure 10. The PINN-evaluated $u(x,t)$ at different timeslices, $t = 0, 0.25, 0.5, 1.0$, from left to right when additive Gaussian errors with $\sigma = 0.5$ is introduced on initial time-slice for when additional regularization in terms of Cole-Hopf transformation is added to the loss function. A single domain PINN used for the top row while the middle and bottom rows show result from 2 and 3 domain cPINNs respectively.

Figure 10(e)–(h) [(i)–(l)] shows the results when we trained the Cole-Hopf regularized PINN with functional and flux continuity imposed at the interface of two [three] sub-domains. These results establish the inadequacy of the Cole-Hopf regularizer in establishing the robustness of the neural architecture and reinforces their tendency to converge to local minimum instead of reaching the intended global minimum.

3. GP based error correction for PINNs

As the previous section illustrates, physics-inspired regularization alone does not eliminate propagation of errors in PINNs. In fact, application of such constraints forces the PINN to converge to a local minimum that satisfies the physics of conservation laws for overfitted boundary conditions and eventually propagates the overfitting across the spatio-temporal domain. In this section, we seek the explore alternate solution to this problem by using smoothing techniques that safeguard the quality of fit by cross-validated regulation of smoothed boundary data. The model of corruption considered in the PDE model for 1D Schrödinger and Burgers' equations is a ubiquitous approximation for many physics processes. In such processes, the spatial evolution of a local field is often expected to be smooth. When the physical data on domain boundary is subject to such errors, it is often convenient to model these measurements as a realization of a stochastic process. For instance, the initial condition in equation (14) can be modeled by a pair of continuous stochastic processes, U_x, V_x where the index representing the spatial coordinate of the PDE domain. The mean and covariance for such processes are given as

$$\begin{aligned}
&\mathbb{E}(U_i) = 2\text{sech}(x = x_i)\\
&\mathbb{E}(V_i) = 0\\
&\text{Cov}(U_i, U_j) = \text{Cov}(V_i, V_j) = \sigma^2 \delta_{ij}.
\end{aligned}$$

To obtain a functional estimate of these stochastic processes, GP Regression [30] is a powerful, nonparametric method. Given the set of samples on the initial timeslice, $\mathcal{U}_B$, the DNN structure in equation (27) is replaced by,

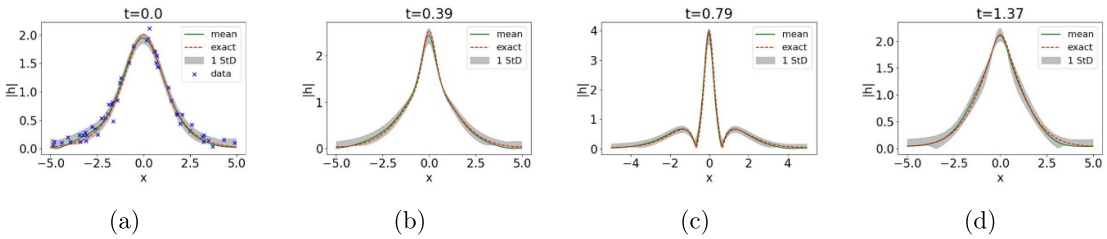


Figure 11. The GP-smoothed PINN-evaluated $|h(x,t)|$ at different timeslices, $t = 0, 0.39, 0.78, 1.37$, from left to right. In this case the training data on the initial timeslice is subject to measurement errors, modeled by a Gaussian random variable with zero mean and a standard deviation of 0.1. GP-based smoothing was used on initial timeslice before training the PINN. The points marked with the blue cross (x) pointer in the leftmost set of plots indicate the samples on the initial timeslice used to fit the GP. The grey band in the subsequent plots represents the uncertainty associated with the PINN-evaluated approximation of $|h(x,t)|$.

$$\tilde{u}(\vec{x}) = \mathbf{NN}_\theta\left(\vec{x}; \hat{\mathcal{U}}_B, \mathcal{U}_C, \mathcal{U}_D\right) \tag{27}$$

where

$$\hat{\mathcal{U}}_B = \{(\vec{x}_i^b, \mathcal{B}[\hat{u}(\vec{x}_i^b)])_{i=1}^{N_b}\} \tag{28}$$

and

$$\hat{u}(\vec{x}_i^b) = \mathrm{GP}\left(\vec{x}_i^b \middle| \left\{(\vec{x}_i^b, u(\vec{x}_i^b))_{i=1}^{N_b}\right\}\right) \tag{29}$$

represents the GP-predicted estimate of the boundary data. The choice of the kernel function, representing the pairwise covariance of observations is given as a sum of RBF and white noise kernels,

$$k(x_i, x_j) = A \exp\left(-\frac{|\vec{x}_i - \vec{x}_j|^2}{2l^2}\right) + \sigma^2 \delta_{ij} \tag{30}$$

where A, l, σ are hyperparameters obtained by maximizing the log-marginal likelihood.

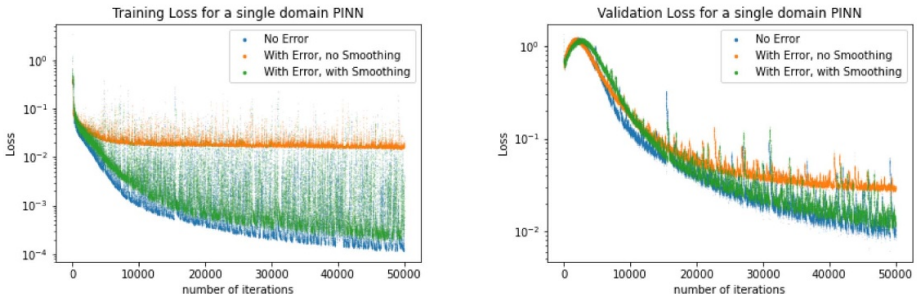
Smoothing techniques are commonly applied in problems where robustness is a desired quality. Compared to other parametric smoothing techniques like fixed order polynominals or smoothing splines, GP regression has often been proved to be more robust against underfitting and overfitting [31, 32]. Robustness guarantees for GPs have been extensively explored in literature [33, 34]. GPs have also been explored in connection with physics-inspired kernel building [35, 36] and found to be effective in predicting physical phenomena like phase transitions in quantum systems [37].

Using a GP-smoothing on the boundary data allows for the PDE solver to regain its performance by training itself over the smoothed data on initial timeslice. Compared to other approaches [38–40] that employ GPs to solve differential equations, our method uniquely harnesses the smoothing interpolating functionality of an GP while exploiting the universal approximator feature of an NN. While GPs with a proper choice of a kernel can be very useful in approximating smooth analytical solutions, their $o(n^3)$ complexity makes them infeasible for optimizing such solutions over a large set of collocation points and such complexity grows significantly with high dimensional problems. However, restricting their use on the domain boundary reduces the complexity by an order of magnitude while almost identically recovering the analytical solution. Figure 11 shows the performance of an GP-smoothed PINN in solving the Schrödinger equation, where the DNN can recover the analytical form despite corruption in initial data.

Since the loss function in equation (5) is not a direct metric of validating the performance of the PINN, the validation loss is measured in terms of *mean squared error (MSE)* loss compared with respect to the analytical solution. In figure 12, we compare the evolution of the loss function during training and the validation MSE loss. It can be seen that GP-smoothed PINN performs almost as well as error-free PINN, and significantly better than a PINN trained with corrupted boundary data but no smoothing.

3.1. Kernel selection for GPs

The choice of kernel for fitting a GP to the boundary data is very important- improper choices can lead to underfitting or overfitting and eventually propagate large errors through the PINN architecture. In order to make the best choice for a kernel, we explored a k-fold cross-validation technique on the initial time-slice



(a) (b)

Figure 12. The training loss function from equation (5) (left) and the MSE validation loss from as a function of number of iterations for a PINN solving.

Table 1. Comparison of average training and validation MSE losses for training and validation data on the domain boundary as given in equation (14) with $\sigma = 0.1$.

kernel	MSE loss on training data	MSE loss on validation data
RBF	0.007 62	0.0110
Matèrn ($\nu = 0.1$)	3.2×10^{-8}	0.0465
Matèrn ($\nu = 1.5$)	0.005 98	0.0116
Matèrn ($\nu = 4.0$)	0.005 88	0.0126
Rational Quadratic	0.007 21	0.0112

data for the 1D Schrödinger equation. The dataset on initial boundary is split into k equal subsets where $k-1$ of them are used for training and one subset is kept aside for validation. Training and validation data are used to optimize the GP hyperparameters. We examine the performance of the optimized GPs for using the RBF kernel, the Matèrn kernel [30] with $\nu = 0.1, 1.5$, and 4.0, and the Rational Quadratic (RQ) kernel [30]. Each kernel is appended with a localized white noise kernel. The average training and validation MSE losses, measured with respect to a fixed set of noise-corrupted sampling points on the initial timeslice (equation (14)), with $k = 10$ for different choices of kernels are summarized in table 1. It can be seen that RBF and RQ kernels have similar performance while the Matèrn kernels tend to overfit.

3.2. Evolution of measurement uncertainty

While a generic PINN fails to recover the physics-motivated evolution of noisy boundary data following a PDE, a GP-smoothed PINN not only can recover the physical evolution but also provide a controlled estimate of uncertainty at every point in the spatio-temporal domain. The uncertainty evaluated by a GP-smoothed PINN is obtained by evaluating the deviation in the NN parameters for $\pm 1\sigma$ variation of the training data on domain boundary

$$\tilde{u}(\vec{x}) \pm \delta\tilde{u}(\vec{x}) = \mathbf{NN}_{\theta \pm \delta\theta}\left(\vec{x}; \hat{\mathcal{U}}_B^{\pm}, \mathcal{U}_C, \mathcal{U}_D\right) \tag{31}$$

where

$$\hat{\mathcal{U}}_B^{\pm} = \{(\vec{x}_i^b, \mathcal{B}[\hat{u}(\vec{x}_i^b) \pm \delta\hat{u}(\vec{x}_i^b)])_{i=1}^{N_b}\}. \tag{32}$$

The uncertainty associated with the boundary data, $\delta\hat{u}$ is obtained from the covariance estimate of the optimized GP. The deviation of the NN parameters, $\delta\theta$ can be obtained from minimizing the loss function evaluated with $\hat{\mathcal{U}}_B^{\pm}$.

$$(\theta \pm \delta\theta)^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{\mathrm{PINN}}(\theta; \hat{\mathcal{U}}_B^{\pm}). \tag{33}$$

Analytical estimate of $\delta\theta^*$ is a computationally intractable task since it requires inversion of the very large Hessian matrix $\frac{\partial^2 \mathcal{L}}{\partial \theta^2}$. However, a rather inexpensive technique is to start with a PINN architecture with parameters θ already optimized for the mean value of the boundary data $\hat{\mathcal{U}}_B$ and re-train the network with the modified boundary data. This reoptimization converges more quickly and provides an estimate of

Algorithm 1. Selection of IPs for SGP.

procedure IPSelect(n_0, M, X, y, ρ)
  Randomly Select X_0, y_0 from X, y with $|X_0| = |y_0| = n_0$
  $k^* = \text{argmax} \log p(y_0|\text{GP}(k, X_0, y_0))$
  for $z \in X - X_0$ **do**
    if $\max\{k^*(z, x_0)|x_0 \in X_0\} < \rho$ **then**
      $X_0 \leftarrow X_0 \cup \{z\}$
      if $|X_0| = M$ **then**
        break
  return X_0

evolution of uncertainty at all points of the space-time domain. The evolution of uncertainty for a GP-smoothed PINN evaluated solution of equation (12) is shown in figure 11. The network was re-trained for an additional 1000 iterations to optimize for the uncertainty bands. In general, the number of additional required to converge for estimating the uncertainty bands depends on the size of the corrupting error, which can be quantitatively estimated from the optimized value of the σ parameter in equation (29).

3.3. Sparse GP (SGP) based error correction

GP-based smoothing can provide robustness for PINNs as shown in the previous section. However, optimizing a GP is an expensive process with a complexity of $o(n^3)$, with n being the number of points considered to optimize the GP. Even though we are restricting the GPs to be optimized only over the domain boundary, this can be still be a major bottleneck for our method for high dimensional problems. As the dimension of domain boundary $\partial D \subset \mathbb{R}^{d-1}$ increases, it will require more and more points on the boundary to satisfy the boundary condition. SGPs have been extensively studied in literature to significantly reduce the complexity for high dimensional problems. A multitude of variants of sparsity inducing GPs have found their applications in the context of sample efficient reinforcement learning [41], deep kriging with big data [42], and variational learning of GPs [43]. We consider a hybrid approach for sparsity inducing smoothing GP on the domain boundary following the algorithm suggested in [44]. to obtain inexpensive selection of IPs. Originally designed for sparse variational GPs (SVGPs), this algorithm is effective in the context of our problem of selecting a smaller subset of IPs on the domain boundary.

The SGP algorithm we use is explained in algorithm 1. The sparsity optimizations for GP is done in two steps. In the first step, a small number of data points (n_0) are randomly taken to optimize the GP hyperparameters. In the second step, additional IPs are chosen from the data based on the kernel distance between the new IP candidate (z) and the already selected set of IPs (X_0). The new IP candidate is included in X_0 if the kernel distance between z and all existing IPs is smaller than some predefined threshold (ρ). To reduce the complexity of this approach, iterative re-optimization of the kernel hyperparameters is avoided and only after the desired set of IPs have been chosen, the GP hyperparameters are finally reoptimized to smooth the corrupted dataset on the domain boundary. The total number of IPs chosen is bounded by $M \leqslant N_{b,t}$. Figure 13 shows how SGPs can be almost equally useful in recovering the Schrödinger field dynamics. When the number of IPs is set too low, e.g. only 10 IPs for both $u(x, 0)$ and $v(x, 0)$, the recovery of physical dynamics is not as satisfactory. However, with a somewhat larger set of IPs including 29 and 20 IPs for $u(x, 0)$ and $v(x, 0)$ respectively, the PINN's performance improves significantly and becomes comparable to that of the full GP-smoothed PINN shown in figure 11.

Table 2 summarizes the validation MSE obtained with different models and compare them with the benchmark model of a vanilla PINN with no errors. While the performance of a PINN significantly deteriorates with the introduction of even modest errors with $\sigma = 0.1$, both GP-smoothed PINN and SVGP-smoothed PINN perform similar to the benchmark model.

We demonstrate the effectiveness of GP and SGP smoothing in recovering the physical field dynamics for 1D Burgers' equation in figures 14(a)–(d) and (e)–(h). The SGP employed 41 IPs on the initial timeslice and shows remarkable performance recovery. We also compare the results from the UQ-PINN architecture proposed in [20] and both GP and SGP smoothed PINNs perform noticeably better than the solution obtained from the UQ-PINN architecture. Table 3 summarizes the validation MSE loss obtained from different PINN architectures and it can be seen that both GP and SGP smoothed recover a similar level of accuracy as observed by the error-free PINN.

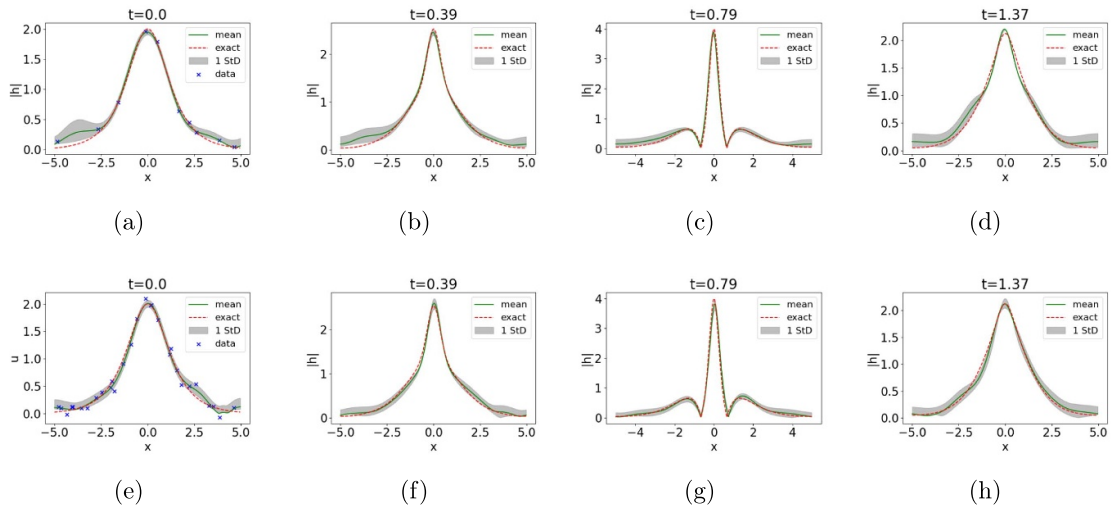


Figure 13. The SGP-smoothed PINN-evaluated $|h(x,t)|$ at different timeslices, $t = 0, 0.39, 0.79, 1.37$, from left to right. In this case the training data on the initial timeslice is subject to measurement errors, modeled by a Gaussian random variable with zero mean and a standard deviation of 0.1. SGP based smoothing was used on initial timeslice before training the PINN using 10 inducing points for each $u(x,0)$ and $v(x,0)$ for the top row and 29 inducing points for $u(x,0)$ and 20 for $v(x,0)$ for the bottom row. The points marked with the blue cross (x) pointer in the leftmost set of plots indicate the samples on the initial timeslice used to train the PINN. The grey band in the subsequent plots represents the uncertainty associated with the PINN-evaluated approximation of $|h(x,t)|$.

Table 2. Comparison of PINNs using different strategies for robustness to solve the 1D nonlinear Schrödinger equation. The introduction of error in the initial condition causes a significant increase in MSE for the standard PINN. GP-smoothing reduces the MSE to nearly as low as the PINN with no error. SGP-smoothing is also effective in reducing error and uses fewer inducing points (IPs). However, if the SGP does not have a sufficient number of IPs the error increases as seen when 10 IPs are used. Multiple domain cPINNs have worse performance. Results quoted for L_1 and L_2 regularizations are taken from the best performance observed over choices of $\lambda \in \{10^{-n}\}_{n=1}^{5}$.

Model	MSE
PINN (no error)	0.0105
PINN ($\sigma = 0.1$)	0.0289
PINN ($\sigma = 0.1$, L_1 regularization with $\lambda = 10^{-4}$)	0.1613
PINN ($\sigma = 0.1$, L_2 regularization with $\lambda = 10^{-4}$)	0.2681
cPINN-2 (no error)	0.2745
cPINN-2 ($\sigma = 0.1$, no smoothing)	0.4782
cPINN-3 (no error)	0.0258
cPINN-3 ($\sigma = 0.1$, no smoothing)	0.4178
GP-smoothed PINN ($\sigma = 0.1$, 50 IPs for u and v)	0.0125
SGP-smoothed PINN ($\sigma = 0.1$, 10 IPs for u and v)	0.0231
SGP-smoothed PINN ($\sigma = 0.1$, 29 and 20 IPs for u and v)	0.0123

4. Additional examples

To demonstrate the effectiveness of GP and SGP smoothing for higher dimensional PDEs, we consider a couple of 2D PDEs in this section.

4.1. 2D heat equation

The 2D heat equation and the corresponding spatio-temporal boundary conditions are given as:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{34}$$

$$u(x,y,0) = 3\sin(\pi x)\sin(\pi y) + \sin(3\pi x)\sin(\pi y) + \Theta_u \epsilon^u \tag{35}$$

$$u(0,y,t) = u(1,y,t) = u(x,0,t) = u(x,1,t) = 0 \tag{36}$$

where the domain boundary is given as $(x,y,t) \in [0,1] \times [0,1] \times [0,0.1]$ and Θ_u is the acceptance function for the noise term in the initial condition. The analytical solution to this equation is given as

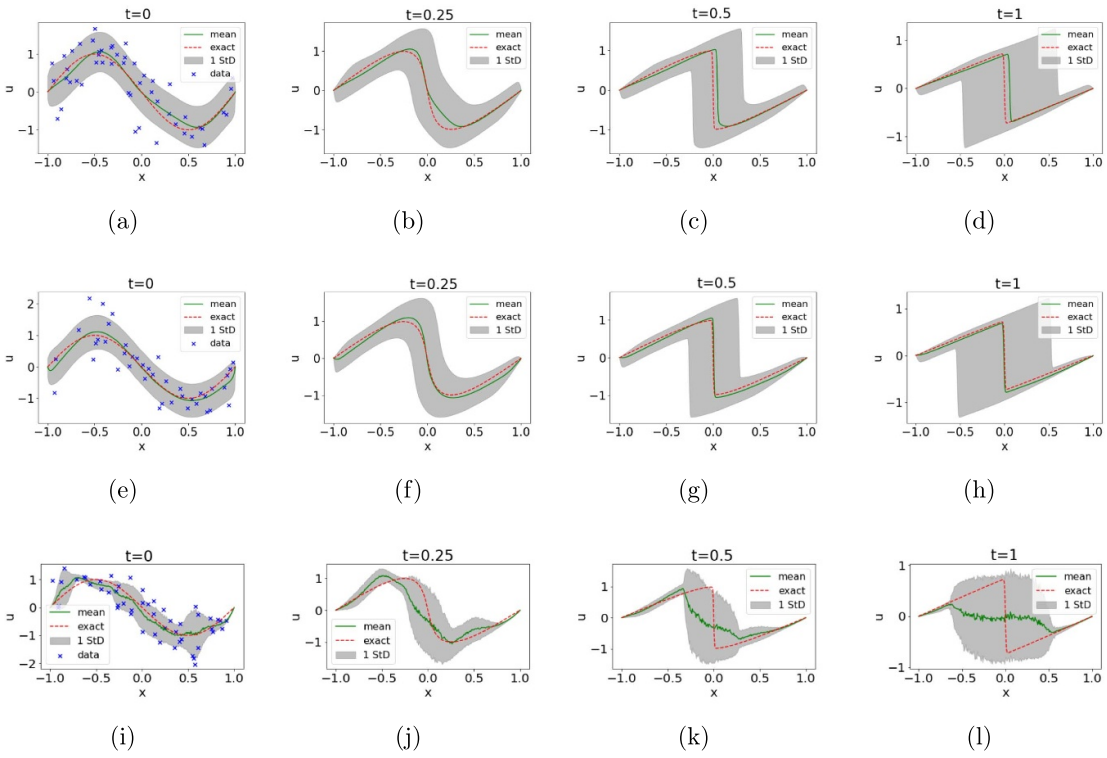
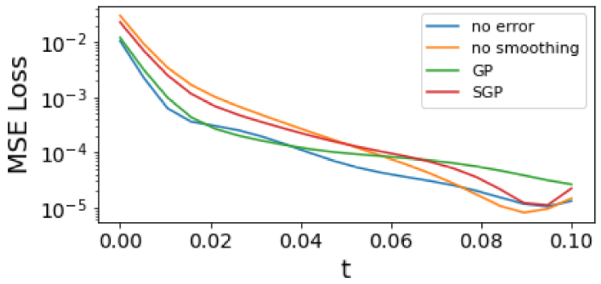


Figure 14. The PINN-evaluated solution to Burgers' equation when measurements on initial timeslice is corrupted with zero-mean, $\sigma = 0.5$ Gaussian noise for GP-smoothed PINN (top), SGP-smoothed PINN (middle), and UQ-PINN [20] (bottom), at time-slices $t = 0, 0.25, 0.51, 1$, from left to right. For the GP- and SGP-smoothed PINNs uncertainty bounds are calculated by retraining the PINN using the initial condition of the GP/SGP mean function plus or minus one standard deviation.

Table 3. Comparison of PINNs using different strategies for robustness to solve the 1D Burgers' equation. The introduction of error in the initial condition causes a significant increase in MSE for the standard PINN. GP-smoothing reduces the MSE to nearly as low as the PINN with no error. SGP-smoothing is also effective in reducing error and uses fewer inducing points (IPs). Results quoted for L_1 and L_2 regularizations are taken from the best performance observed over choices of $\lambda \in \{10^{-n}\}_{n=1}^{5}$.

Model	MSE
PINN (no error)	0.0116
PINN ($\sigma = 0.5$)	0.1982
PINN ($\sigma = 0.1$, L_1 regularization with $\lambda = 10^{-4}$)	0.0392
PINN ($\sigma = 0.1$, L_2 regularization with $\lambda = 10^{-4}$)	0.0293
PINN ($\sigma = 0.5$, Cole-Hopf regularizer)	0.1125
cPINN-2 (no error)	0.0161
cPINN-2 ($\sigma = 0.5$, no smoothing)	0.0834
cPINN-2 ($\sigma = 0.5$, Cole-Hopf regularizer)	0.0891
cPINN-3 (no error)	2.782×10^{-5}
cPINN-3 ($\sigma = 0.5$, no smoothing)	0.0854
cPINN-3 ($\sigma = 0.5$, Cole-Hopf regularizer)	0.0329
UQ-PINN [20] ($\sigma = 0.5$)	0.1248
GP-smoothed PINN ($\sigma = 0.5$, 50 IPs)	0.0384
SGP-smoothed PINN ($\sigma = 0.5$, 41 IPs)	0.0080

$u(x,y,t) = 3\sin(\pi x)\sin(\pi y)e^{-2\pi^2 t^2} + \sin(3\pi x)\sin(\pi y)e^{-10\pi^2 t^2}$. An MLP with four hidden layers, each with 256 nodes, has been used. The physics is enforced with $Nc = 50000$ collocation points. 64 points are chosen on each of the four spatial boundaries and 1024 points on the initial timeslice for the initial condition, giving a total of $Nb = 1280$ points on the spatio-temporal boundary. Like the previous examples, the loss function is constructed according to equation (5) with $\alpha_{()} = 1.0$. For the SGP process, the IPs are chosen from the pool of 1024 points on the initial time slice according to Algorithm 1 with the number of IPs bounded by $M = 768$. The models are trained for 200 00 epochs with ADAM optimizer with a learning rate of 10^{-3}.



Model	MSE
PINN (no error)	7.6e-4
PINN ($\sigma = 1.0$)	2.4e-3
GP-PINN ($\sigma = 1.0$, 1024 IPs)	9.2e-4
SGP-PINN ($\sigma = 1.0$, 602 IPs)	1.8e-3

Figure 15. The time evolution of the MSE loss for different models used in solving the 2D heat equation. Except the vanilla PINN (no error), all models were trained with data sampled from the initial timeslice corrupted with additive Gaussian noise with zero mean and $\sigma = 1.0$. MSE error evaluated over 50k points chosen over the entire spatio-temporal domain for the different models is given in the accompanying table.

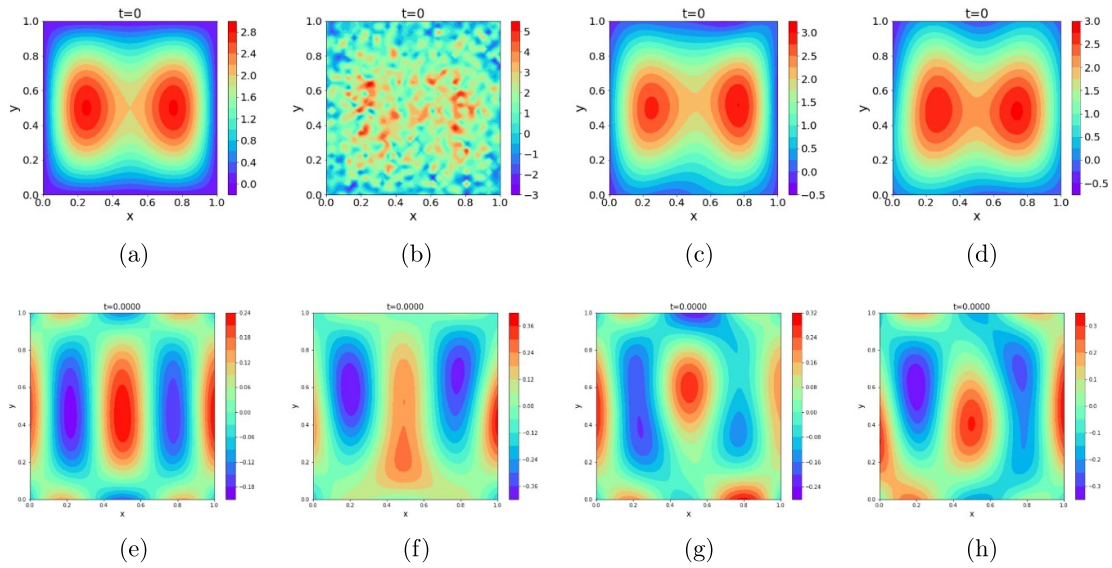


Figure 16. The initial condition at $t = 0$ for 2D heat equation for (a) error-free PINN, (b) noisy PINN without smoothing, (c) noisy PINN with GP smoothing, (d) noisy PINN with SGP smoothing. Except the vanilla PINN (no error), all models were trained with data sampled from the initial timeslice corrupted with additive Gaussian noise with zero mean and $\sigma = 1.0$. The figures in the bottom row show the error in PINN-evaluated solution at the initial timeslice for the corresponding architecture in the top row.

As shown in figure 15, GP-smoothing recovers the performance of the error-free PINN and SGP also considerably brings down the MSE when compared to that of the PINN trained with noisy data without any smoothing applied. We can see the initial condition each PINN architecture is trained with along with the point-wise error estimate in the PINN's solution for different models. The smoothing effect on the initial timeslice can be seen in figure 16, where we can see that while the noisy initial condition almost completely obliterates the distributive feature of $u(x, t = 0)$, smoothing with GP or SGP allows its significant recovery. This translates into better convergence to actual solution for the latter couple of models on both initial and latter timeslices (figure 17).

4.2. 2D Burgers' equation

The two dimensional Burgers' equation is given by the following pair of PDEs-

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \end{aligned} \tag{37}$$

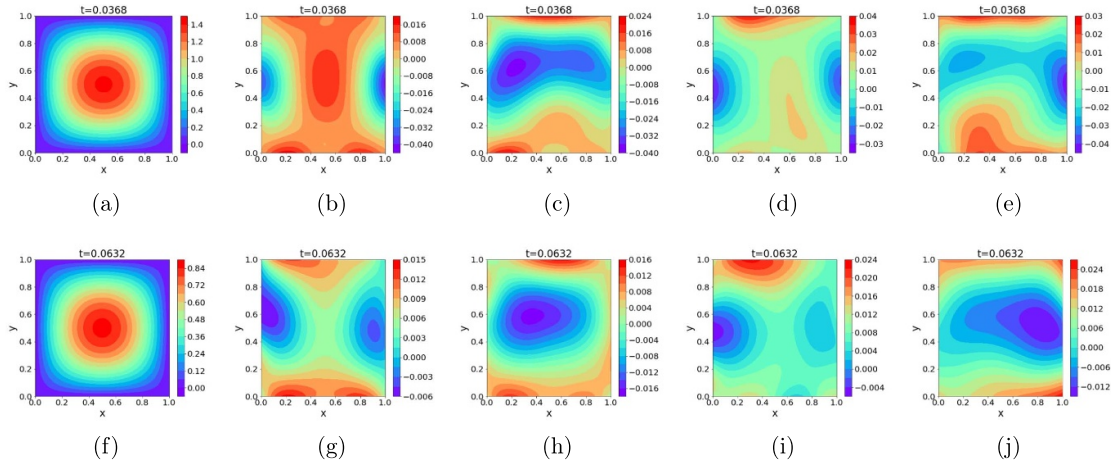
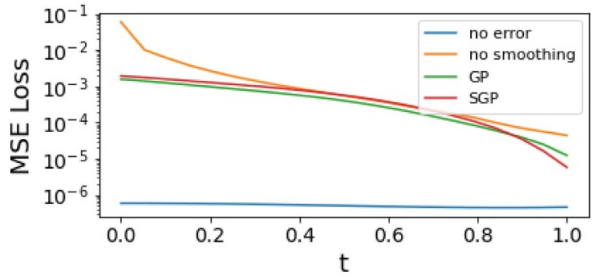


Figure 17. The leftmost column shows the exact solution for 2D heat equation at $t = 0.0368$ (top row) and $t = 0.0632$ (bottom row). The remaining columns show the point-wise error in PINN-evaluated solution for noise-free PINN (second column), noisy PINN without smoothing (third column), GP-PINN (fourth column), and SGP-PINN (final i.e. fifth column). Except the vanilla PINN (no error), all models were trained with data sampled from the initial timeslice corrupted with additive Gaussian noise with zero mean and $\sigma = 1.0$.



Model	MSE
PINN (no error)	5.16e-7
PINN ($\sigma = 0.5$)	4.4e-2
GP-PINN ($\sigma = 0.5$, 1024 IPs)	5.3e-4
SGP-PINN ($\sigma = 0.5$, 512 IPs)	7.0e-4

Figure 18. The figure demonstrates the time evolution of the MSE loss for different models used in solving the 2D Burgers' equation. Except the vanilla PINN (no error), all models were trained with data sampled from the initial timeslice corrupted with additive Gaussian noise with zero mean and $\sigma = 0.5$. MSE error evaluated over 50k points chosen over the entire spatio-temporal domain for the different models is given in the accompanying table.

where we consider $\nu = \frac{0.01}{\pi}$ and train the network to learn the following analytical solution [45, 46]-

$$u(x, y, t) = \frac{3}{4} - \frac{1}{4\left(1 + \exp\left(\frac{-t-4x+4y}{32\nu}\right)\right)} \tag{38}$$

$$v(x, y, t) = \frac{3}{4} + \frac{1}{4\left(1 + \exp\left(\frac{-t-4x+4y}{32\nu}\right)\right)}. \tag{39}$$

The domain boundary is chosen as $(x, y, t) \in [0, 1] \times [0, 1] \times [0, 1]$. The network is trained with the initial condition sampled from the functions $u(x, y, 0) + \Theta_u \epsilon^u$ and $v(x, y, 0) + \Theta_v \epsilon^v$ respectively for u and v where Θ_u and Θ_v are the acceptance functions for the noise terms in the initial condition. The spatial boundary conditions are obtained from plugging in the boundary coordinates in the analytical solution given in Equations (38) and (39). An MLP with four hidden layers, each with 256 nodes, has been used to simultaneously predict the two fields. The physics is enforced with $Nc = 50000$ collocation points. We choose 64 points on each of the four spatial boundaries and 1024 points on the initial timeslice to enforce the spatio-temporal boundary condition with a total $Nb = 1280$ measurements. The choice of loss function and optimizer follows the example of the previous examples. A pool of 1024 uniformly sampled points on the initial timeslice are used for SGP and IPs are chosen according to Algorithm 1 with the number of IPs bounded by $M = 768$. The resulting performances of the four models after training for 20 000 epochs are shown in figure 18. As we can see from the accompanying table in figure 18, the noise-free PINN replicates

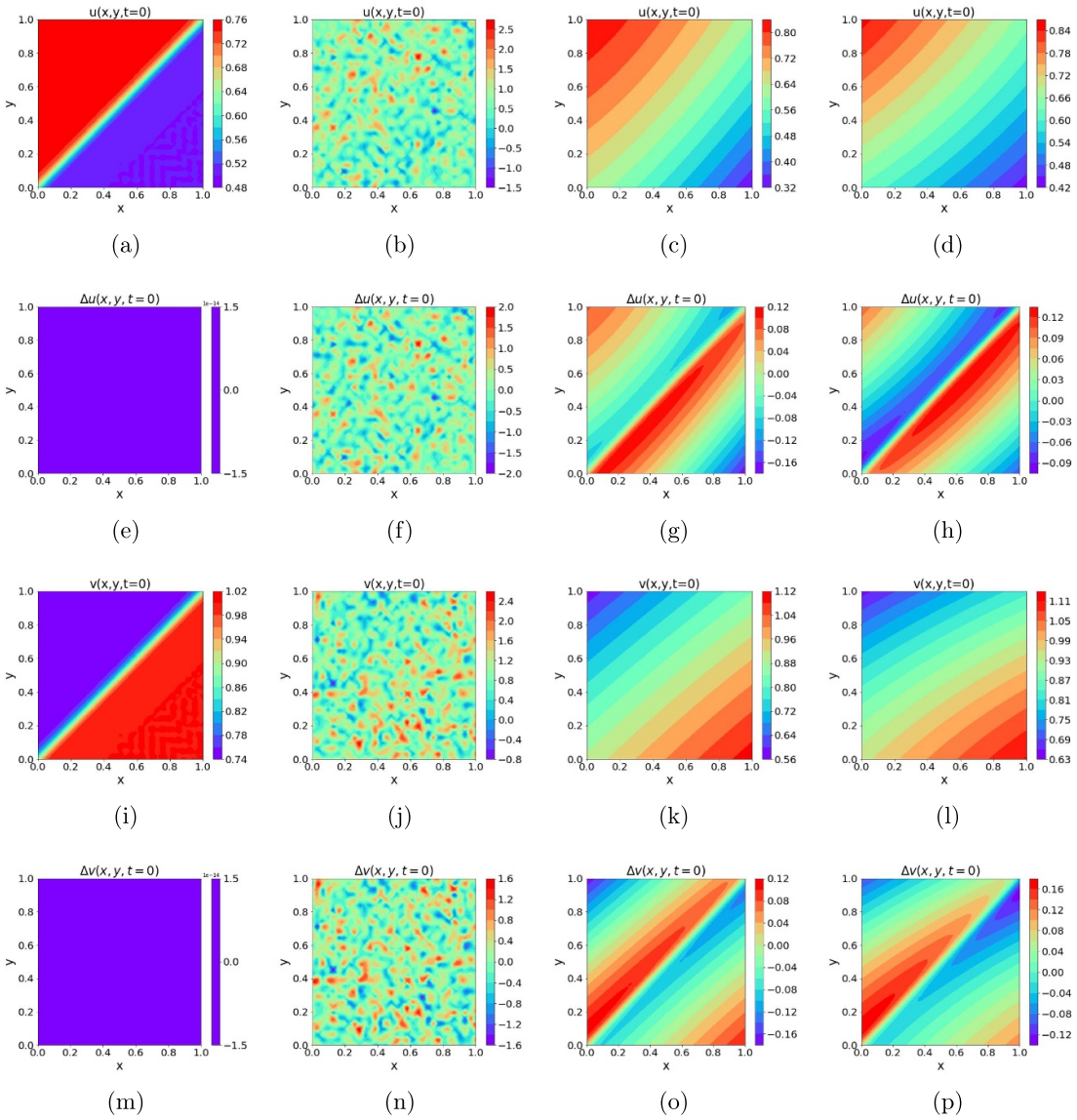


Figure 19. The initial condition $u(x, y, t = 0)$ for 2D Burgers' equation for (a) error-free PINN, (b) noisy PINN without smoothing, (c) noisy PINN with GP smoothing, (d) noisy PINN with SGP smoothing. The figures in the second row show the error in PINN-evaluated solution at the initial timeslice for the corresponding architecture in the top row. The final two rows show the equivalent distributions for the $v(x, y, t)$ field. Except the vanilla PINN (no error), all models were trained with data sampled from the initial timeslice corrupted with additive Gaussian noise with zero mean and $\sigma = 0.5$.

the analytical solution almost perfectly and when error is introduced, similar level of performance cannot be retrieved. However, the MSE is noticeably reduced by the smoothing performed by GP and SGP.

One of the noticeable aspects of the 2D Burgers' equations is the presence of a shockwave front at $4y - 4x = t$ around where both fields experience rather sharp, yet continuous gradients. We show the exact initial condition used to train the noise-free PINN in figures 19(a) and (i). When corrupted with noise and left unsmoothened, the shockwave feature is almost completely lost and as can be seen from figure 20 (third column from the left), the network struggles to retrieve the shockwave front for latter timeslices as well. On the other hand, while GP and SGP to some extent recovers the initial field distributions, the gradients near the shockwave front are oversmoothed. It should be noted that this oversmoothing is not due to some limitation of GP or SGP itself, but rather the choice of samples used to train these processes. Being agnostic to the physical distribution, the enforcing points on the initial timeslice are uniformly sampled, which led to an under-representation of sharp gradients near the shockwave front. As a consequence to the missing perception of sharpness around the shockwave front, the GP-PINN and SGP-PINN accumulate their largest deviations in the latter timeslices around the shockwave front, as seen in figure 20 (fourth and fifth columns from the left). The opposite signs of the errors on the two sides of the shockwave front represent that the PINN-reconstructed solution after GP or SGP smoothing has a more smoothly shifting wavefront.

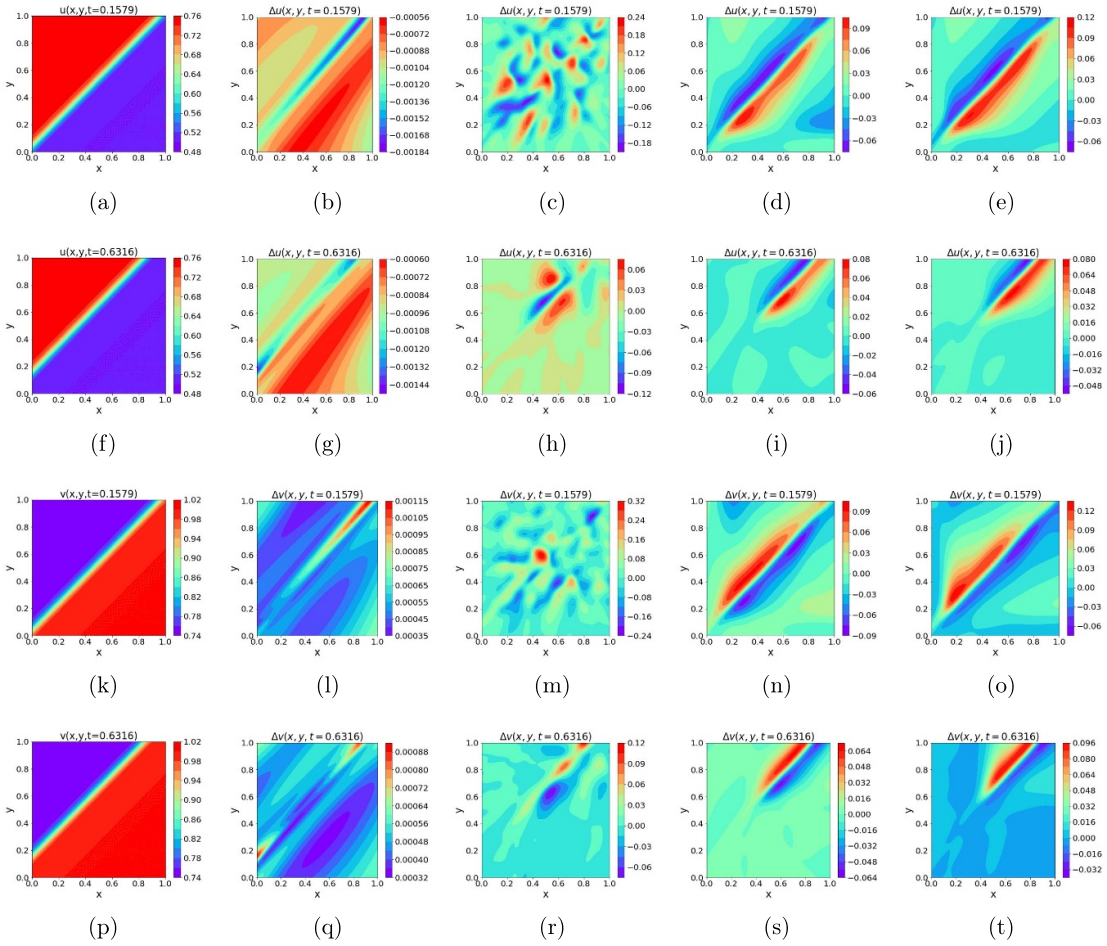


Figure 20. The leftmost column shows the exact solution for 2D Burgers' equation at $t = 0.1579$ (first and third row from top for $u(x, y, t)$) and $t = 0.6316$ (second and fourth row from top for $v(x, y, t)$). The remaining columns show the point-wise error in PINN-evaluated solution for noise-free PINN (second column), noisy PINN without smoothing (third column), GP-PINN (fourth column), and SGP-PINN (final i.e. fifth column). Except the vanilla PINN (no error), all models were trained with data sampled from the initial timeslice corrupted with additive Gaussian noise with zero mean and $\sigma = 0.5$.

5. Conclusion

As it often happens, measurements associated physical processes are subject to errors. When these measurements are used to learn the evolution of a system respecting some underlying physics dictated by a PDE using NNs, these errors can significantly distort the predicted behavior via nonlinear propagation of errors. In this paper, we explored the behavior of a PINN when it is trained with noise-corrupted datasets. Our work shows that deep PDE-solvers can be subject to overfitting and dynamically propagating errors observed on the domain boundaries even when physics-inspired regularizers are introduced to constrain the solution. To circumvent this issue, we proposed GP-smoothed deep network that can help recover the system's behavior over a finite space-time domain while providing a controlled prediction and bounded uncertainty. We further showed that the computational complexity of fitting an GP can be significantly reduced by incorporating sparsely choosing IPs for SGPs. This opens up opportunities to explore uncertainty propagation in predictive estimation using cPINNs or cPINN-like architectures as well as learning an optimal policy of selecting sparsely chosen IPs.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/CVC-Lab/RobustPINNs.

Acknowledgment

The research for C B and L M was supported in part by NIH-R01GM117594, by the Peter O'Donnell Foundation, and in part from a grant from the Army Research Office accomplished under Cooperative Agreement No. W911NF-19-2-0333. The work of L M was additionally supported by the Moncreif Summer Undergraduate Internship Program. The work of T A and A R is supported by U.S. Department of Energy, Office of High Energy Physics under Grant No. DE-SC0007890. Studies performed by A R utilize resources supported by the National Science Foundation's Major Research Instrumentation program, Grant 1725729, as well as the University of Illinois at Urbana-Champaign

ORCID iD

Avik Roy https://orcid.org/0000-0002-0116-1012

References

[1] Noé F, Tkatchenko A, Müller K-R and Clementi C 2020 Machine learning for molecular simulation *Annu. Rev. Phys. Chem.* **71** 361–90
[2] Puscasu G, Codres B, Stancu A and Murariu G 2009 Nonlinear system identification based on internal recurrent neural networks *Int. J. Neural Syst.* **19** 115–25
[3] Carleo G, Nomura Y and Imada M 2018 Constructing exact representations of quantum many-body systems with deep neural networks *Nat. Commun.* **9** 5322
[4] Benzi M 2002 Preconditioning techniques for large linear systems: a survey *J. Comput. Phys.* **182** 418–77
[5] Grossmann C, Roos H-Gorg and Stynes M 2007 *Numerical Treatment of Partial Differential Equations* vol 154 (Berlin: Springer)
[6] Dissanayake M W M G and Phan-Thien N 1994 Neural-network-based approximations for solving partial differential equations *Commun. Numer. Methods Eng.* **10** 195–201
[7] Aarts L P and Van Der Veer P 2001 Neural network method for solving partial differential equations *Neural Process. Lett.* **14** 261–71
[8] Hayati M and Karami B 2007 Feedforward neural network for solving partial differential equations *J. Appl. Sci.* **7** 2812–7
[9] Baydin A G, Pearlmutter B A, Radul A A and Siskind J M 2018 Automatic differentiation in machine learning: a survey *J. Mach. Learn. Res.* **18** 5595–637
[10] Lagaris I E, Likas A and Fotiadis D I 1998 Artificial neural networks for solving ordinary and partial differential equations *IEEE Trans. Neural Netw.* **9** 987–1000
[11] Ricky T Q Chen Y R, Bettencourt J and Duvenaud D 2018 Neural ordinary differential equations *Advances in Neural Information Processing Systems*
[12] Pontryagin L S, Mishchenko E F, Boltyanskii V G and Gamkrelidze R V 1987 *Mathematical Theory of Optimal Processes* (New York: Gordon and Breach Science Publishers)
[13] Yıldız Çağatay, Heinonen M and Lähdesmäki H 2019 Ode2vae: Deep generative second order odes with bayesian neural networks (arXiv:1905.10994)
[14] Raissi M, Perdikaris P and Karniadakis G E 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707
[15] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Netw.* **2** 359–66
[16] Pinkus A 1999 Approximation theory of the mlp model in neural networks *Acta Numer.* **8** 143–95
[17] Pang G, Lu L and Karniadakis G E 2019 fpinns: Fractional physics-informed neural networks *SIAM J. Sci. Comput.* **41** A2603–26
[18] Yang L, Meng X and Karniadakis G E 2020 B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data (arXiv:2003.06097)
[19] Jagtap A D, Kawaguchi K and Karniadakis G E 2020 Adaptive activation functions accelerate convergence in deep and physics-informed neural networks *J. Comput. Phys.* **404** 109136
[20] Yang Y and Perdikaris P 2019 Adversarial uncertainty quantification in physics-informed neural networks *J. Comput. Phys.* **394** 136–52
[21] Meng X, Li Z, Zhang D and Karniadakis G E 2020 Ppinn: parareal physics-informed neural network for time-dependent pdes *Comput. Methods Appl. Mech. Eng.* **370** 113250
[22] Jagtap A D, Kharazmi E and Karniadakis G E 2020 Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems *Comput. Methods Appl. Mech. Eng.* **365** 113028
[23] Shin Y 2020 On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes *Commun. Comput. Phys.* **28** 2042–74
[24] Kharazmi E, Zhang Z and Karniadakis G E 2019 Variational physics-informed neural networks for solving partial differential equations (arXiv:1912.00873)
[25] Fujiwara K and Miyazaki H 2014 The derivation of conservation laws for nonlinear schrodinger equations with a power type nonlinearity (arXiv:1407.5282)
[26] Watanabe S, Miyakawa M and Yajima N 1978 Method of conservation laws for solving nonlinear schrodinger equation *J. Phys. Soc. Japan* **46** 1653–9
[27] Barrett J 2013 The Local Conservation Laws of the Nonlinear Schrodinger Equation *PhD Thesis* Oregon State University
[28] Cole J D 1951 On a quasi-linear parabolic equation occurring in aerodynamics *Q. Appl. Math.* **9** 225–36
[29] Hopf E 1950 The partial differential equation $u_t + uu_x = \mu u_{xx}$ *Commun. Pure Appl. Math.* **3** 201–30
[30] Williams C K and Rasmussen C E 2006 *Gaussian Processes for Machine Learning* vol 2 (Cambridge, MA: MIT press)
[31] Schulz E, Speekenbrink M and Krause A 2018 A tutorial on gaussian process regression: modelling, exploring and exploiting functions *J. Math. Psychol.* **85** 1–16
[32] Gramacy R B 2020 *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences* (Chapman and Hall: CRC)

[33] Cardelli L, Kwiatkowska M, Laurenti L and Patane A 2019 Robustness guarantees for bayesian inference with gaussian processes *Proc. AAAI Conf. on Artificial Intelligence* vol 33 pp 7759–68
[34] Blaas A, Patane A, Laurenti L, Cardelli L, Kwiatkowska M and Roberts S 2020 Adversarial robustness guarantees for classification with gaussian processes *Int. Conf. on Artificial Intelligence and Statistics* (PMLR) pp 3372–82
[35] Duvenaud D K, Nickisch H and Rasmussen C 2011 Additive gaussian processes *Advances in Neural Information Processing Systems* vol 24
[36] Duvenaud D, Lloyd J, Grosse R, Tenenbaum J and Zoubin G 2013 Structure discovery in nonparametric regression through compositional kernel search *Int. Conf. on Machine Learning* (PMLR) pp 1166–74
[37] Vargas-Hernández R A, Sous J, Berciu M and Krems R V 2018 Extrapolating quantum observables with machine learning: inferring multiple phase transitions from properties of a single phase *Phys. Rev. Lett.* **121** 255702
[38] Graepel T 2003 Solving noisy linear operator equations by gaussian processes: application to ordinary and partial differential equations *Proc. 20th Int. Conf. on Int. Conf. on Machine Learning* (*ICML*) vol 3 pp 234–41
[39] Raissi M, Perdikaris P and Karniadakis G E 2018 Numerical gaussian processes for time-dependent and nonlinear partial differential equations *SIAM J. Sci. Comput.* **40** A172–98
[40] Chen Y, Hosseini B, Owhadi H and Stuart A M 2021 Solving and learning nonlinear pdes with gaussian processes (arXiv:2103.12959)
[41] Grande R, Walsh T and How J 2014 Sample efficient reinforcement learning with gaussian processes *Proc. of the 31st Int. Conf. on Machine Learning*
[42] Gadd C, Heinonen M, Lähdesmäki H and Kaski S 2020 Sample-efficient reinforcement learning using deep gaussian processes (arXiv:2011.01226)
[43] Tran D, Ranganath R and Blei D M 2016 The variational gaussian process *4th Int. Conf. on Learning Representations* (*ICLR 2016*)
[44] Galy-Fajou T and Opper M 2020 Adaptive inducing points selection for gaussian processes *Continual Learning Workshop of ICML*
[45] Fletcher C A J 1983 Generating exact solutions of the two-dimensional burgers' equations *Int. J. Numer. Methods Fluids* **3** 213–16
[46] Zhu H, Shu H and Ding M 2010 Numerical solutions of two-dimensional burgers' equations by discrete adomian decomposition method *Comput. Math. Appl.* **60** 840–8