

Characterizing Possible Failure Modes in Physics-Informed Neural Networks

Aditi S. Krishnapriyan^{*,1,2}, Amir Gholami^{*,2},
Shandian Zhe³, Robert M. Kirby³, Michael W. Mahoney^{2,4}
{aditik1, amirgh, mahoneymw}@berkeley.edu, {zhe, kirby}@cs.utah.edu

Abstract

Recent work in scientific machine learning has developed so-called physics-informed neural network (PINN) models. The typical approach is to incorporate physical domain knowledge as soft constraints on an empirical loss function and use existing machine learning methodologies to train the model. We demonstrate that, while existing PINN methodologies can learn good models for relatively trivial problems, they can easily fail to learn relevant physical phenomena even for simple PDEs. In particular, we analyze several distinct situations of widespread physical interest, including learning differential equations with convection, reaction, and diffusion operators. We provide evidence that the soft regularization in PINNs, which involves differential operators, can introduce a number of subtle problems, including making the problem ill-conditioned. Importantly, we show that these possible failure modes are not due to the lack of expressivity in the NN architecture, but that the PINN’s setup makes the loss landscape very hard to optimize. We then describe two promising solutions to address these failure modes. The first approach is to use curriculum regularization, where the PINN’s loss term starts from a simple PDE regularization, and becomes progressively more complex as the NN gets trained. The second approach is to pose the problem as a sequence-to-sequence learning task, rather than learning to predict the entire space-time at once. Extensive testing shows that we can achieve up to 1-2 orders of magnitude lower error with these methods as compared to regular PINN training.

1 Introduction

Partial differential equations (PDEs) are commonly used to describe different phenomena in science and engineering. These PDEs are often derived by starting from governing first principles (e.g., conservation of mass or energy). It is typically not possible to find analytical solutions to these PDEs for many real-world settings. Thus, many different numerical methods (e.g., the finite element method [1], pseudo-spectral methods [2], etc.) have been introduced to approximate their solutions/behavior. However, these PDEs can be quite complex for several settings (e.g., turbulence simulations), and numerical integration techniques, which typically update and improve a candidate solution iteratively until convergence, are often quite computationally expensive. Motivated by this—as well as the increasing quantities of data available in many scientific and engineering applications—there has been recent interest in developing machine learning (ML) approaches to find the solution of the underlying PDEs (and/or work in tandem with numerical solutions). As a result, the area

*Equal contribution. ¹Lawrence Berkeley National Lab, ²University of California Berkeley, ³University of Utah, ⁴International Computer Science Institute

of Scientific Machine Learning (SciML)—which aims to couple traditional scientific mechanistic modeling (typically, differential equations) with data-driven ML methodologies (most recently, neural network training)—has emerged. In this vein, there have been a number of ML approaches to incorporate scientific knowledge into such problems while keeping the automatic, data-driven estimates of the solution [3–6].

A recent line of work involves Physics-Informed Neural Network (PINN) models, which aims to incorporate physical domain knowledge as soft constraints on an empirical loss function, that is then optimized using existing ML training methodologies. To some degree, PINNs are an example of “grafting together” domain-driven models and data-driven methodologies. However, there are important subtleties with this, and we identify several possible failure modes with a naive approach. We then illustrate possible directions for addressing these failure modes.

Background and problem overview. Many of the problems with a PDE constraint fit the following abstraction:

$$\mathcal{F}(u(x, t)) = 0, \quad x \in \Omega \subset \mathbb{R}^d, \quad t \in [0, T], \quad (1)$$

where \mathcal{F} is a differential operator representing the PDE, $u(x, t)$ is the state variable (i.e., parameter of interest), x/t denote space/time, T is the time horizon, and Ω is the spatial domain. Since \mathcal{F} is a differential operator, in general one must specify appropriate boundary and/or initial conditions to ensure the existence/uniqueness of a solution to Eq. 1. In the context of PDEs, \mathcal{F} can be taxonomized into a parabolic, hyperbolic, or elliptic differential operator [7]. Quintessential examples of \mathcal{F} include: the convection equation (a hyperbolic PDE), where $u(x, t)$ could model fluid movement, e.g., air or some liquid, over space and time; the diffusion equation (a parabolic PDE), where $u(x, t)$ could model the temperature distribution over space and time; and the Laplace equation (an elliptic PDE), where $u(x)$ could model a steady-state diffusion equation, in the limit as $t \rightarrow \infty$.

One possible data-driven approach is to incorporate domain information by applying Eq. 1 as a hard constraint when training a NN on the data. This can be formulated as the following constrained optimization problem,

$$\min_{\theta} \mathcal{L}(u) \quad \text{s.t.} \quad \mathcal{F}(u) = 0, \quad (2)$$

where $\mathcal{L}(u)$ is the data-fit term (including initial/boundary conditions), and where \mathcal{F} is a constraint on the residual of the PDE system under consideration (i.e., the “physics” knowledge in the equation itself). As mentioned before, for many practical use cases, it is not possible to derive closed form solutions for these problems, and it is often quite difficult to solve problems of the form of Eq. 2, with $\mathcal{F}(u)$ as a hard constraint.

Another data-driven approach is to impose the constraint as a “soft constraint” on the outputs of the NN model,

$$\min_{\theta} \mathcal{L}(u) + \lambda_{\mathcal{F}} \mathcal{F}(u), \quad (3)$$

$$\mathcal{L}(u) = \mathcal{L}_{u_0} + \mathcal{L}_{u_b}. \quad (4)$$

Here, L_{u_0} and L_{u_b} measure the misfit of the NN prediction and the initial/boundary conditions (which are pre-specified/given as input to the problem), and θ denotes the NN parameters (which takes (x, t) , and possibly other quantities, as inputs and then outputs $u(x, t)$). Furthermore, $\lambda_{\mathcal{F}}$ is a regularization parameter that controls the emphasis on the PDE based residual (which we ideally

want to be zero). The goal is then to use ML methodologies (stochastic optimization, etc.) to train this NN model to minimize the loss in Eq. 3. In particular, the NN is trained to minimize this modified loss function, where the modification is to penalize the violations of $\mathcal{F}(u)$ for some $\lambda_{\mathcal{F}} \geq 0$. However, even with a large training dataset, this approach does not guarantee that the NN will obey the conservation/governing equations in the constraint Eq. 1. In many SciML problems, these sorts of constraints on the system matter, as they correspond to physical mechanisms of the system. For example, if the conservation of energy equation is only approximately satisfied, then the system being simulated may behave qualitatively differently or even result in unrealistic solutions.

We should also note that this approach of incorporating physics-based regularization, where the regularization constraint, $\mathcal{L}_{\mathcal{F}}$, corresponds to a differential operator, is *very* different than incorporating much simpler norm-based regularization (such as L_1 or L_2 regularization), as is common in ML more generally. Here, the regularization operator, $\mathcal{L}_{\mathcal{F}}$ is non-trivially structured—it involves a differential operator that could actually be ill-conditioned. Moreover, $\mathcal{L}_{\mathcal{F}}$ corresponds to actual physical quantities, and there is often an important distinction between satisfying the constraint exactly versus satisfying the constraint approximately (the soft constraint approach doing only the latter).

Main contributions. The contributions of this paper are as follows:

- We analyze PINN models on simple, yet physically relevant, problems of convection, reaction, and reaction-diffusion. We find that the vanilla/regular PINN approach only works for very easy parameter regimes (i.e., small PDE coefficients), but that it fails to learn relevant physics in even moderately more challenging physical regimes, even for problems that have simple closed-form analytical solutions. For many cases, the vanilla PINN approach achieves almost 100% error, as compared to the ground truth solution, even after extensive hyperparameter tuning. (See §3 for details.)
- We analyze the loss landscape of trained PINN models and find that adding/increasing the PDE-based soft constraint regularization ($\mathcal{L}_{\mathcal{F}}$ in Eq. 3) makes it *more* complex and harder to optimize, especially for cases with non-trivial coefficients. We also study how the loss landscape changes as the regularization parameter ($\lambda_{\mathcal{F}}$) is changed. We find that reducing the regularization parameter can help alleviate the complexity of the loss landscape, but this in turn leads to poor solutions with high errors that do not satisfy the PDE/constraint. (See §4 for details.)
- We demonstrate that the NN architecture has the capacity/expressivity to find a good solution, showing that these problems are not due to the limited capacity of the NN architecture. Instead, we argue that the failure is due to optimization difficulties associated with the PINN’s soft PDE constraint. (See §5 for details.)
- We propose two paths forward to address these failure modes through (i) curriculum regularization and (ii) posing the learning problem as a sequence-to-sequence learning task. First, in curriculum regularization, we start by imposing the PDE constraint ($\mathcal{L}_{\mathcal{F}}$) with small coefficients, which are progressively increased to the target problem’s settings as the model gets trained. This gives the NN an opportunity to first train with *easier* constraints, before it is exposed to the target constraint which could be hard to optimize from the beginning. Second, we show that changing the learning problem to a sequence-to-sequence learning problem can reduce the

PINN error, again without any change to the NN architecture. In this setup, the NN is trained on a time segment, instead of the full space-time, which could be more difficult to learn. The task is then to predict the solution and reduce the loss only over smaller time segments. We extensively test both approaches and show that they can reduce the error by up to 1-2 orders of magnitude as compared to regular PINN training, and in many cases can better capture “sharp” features in the solution. (See §5 for details.)

- We have open sourced our framework [8] which is built on top of PyTorch both to help with reproducibility and also to enable other researchers to extend the results.

2 Related work

There is a large body of related work, and here we briefly discuss two of the most related lines of work.

Machine learning and PDEs. ML approaches for PDE problems have been increasing rapidly in recent years [9, 10]. A number of tools and methodologies now exist to solve scientific problems by combining ML and domain insights [11–15]. As mentioned earlier, a popular approach to combine ML and physical knowledge is to include aspects of the PDE term as part of the optimization process via regularization. A notable aspect of such an approach is that the NN can be trained only on data that comes from the governing equation(s) itself (though additional data can be included as well, if available), i.e., with a relatively small amount of data. This has garnered interest and shown successful results in a wide variety of science and engineering problems and applications [16–22].

However, there have also been issues observed with this formulation. For example, it did not work well for stiff ordinary differential equations (ODEs) describing chemical kinetics [23] or for certain fluid flow problems [24]. Furthermore, PINN models have been analyzed in the context of neural tangent kernels (i.e., towards the infinite width limit) to study their convergence [25, 26]. This work found some cases where the model failed (such as when the target function exhibits “high frequency features”) and showed some preliminary solutions via the lens of the neural tangent kernel. It has been argued that some of these problems may be due an imbalance in back-propagated gradients in the loss function during training, and a learning-rate annealing scheme has been proposed to mitigate this [27].

Physical priors and constraints in NNs. Imposing physical priors and constraints on NN systems is common in SciML problems, as a way to try to enforce a property of interest. Some approaches have focused on embedding specialized physical constraints into NNs, such as conservation of energy or momentum [28, 29] or multiscale features [30]. While methods focusing on constraining the output of the NN are more common, it is difficult to enforce such constraints exactly in ML settings. Previous work has tried to impose hard constraints in ML (both within the context of SciML and otherwise) [31–35], although this can be computationally expensive, and does not guarantee better results or convergence.

3 Possible failure modes for physics-informed neural networks

In this section, we highlight several examples where the PINN formulation defined in Eq. 3 does not predict the solution well. We first demonstrate this with two different types of simple, canonical PDE/ODE systems which have simple analytical solutions: convection (§3.1), and reaction (§3.2). We then also include a diffusion component by looking at the reaction-diffusion problem (§3.3). Note that the convection problem has a linear PDE constraint, and reaction/reaction-diffusion problems both have non-linear PDE terms.¹ We show that PINNs can only learn simple problems with very small parameter regimes (e.g., small convection or reaction coefficients). We demonstrate that these models fail to learn the relevant physical phenomena for non-trivial cases (e.g., relatively larger coefficients). As we will see, while adding the physical constraint as a soft regularization may be easier to deploy and optimize with existing unconstrained optimization methods, it does come with trade-offs, including that in many cases the optimization problem becomes much more difficult to solve.

Experiment setup. We study both linear and non-linear PDEs/ODEs, and we vary the convection, reaction, and diffusion coefficients for each problem (hereafter, we refer to these as *PDE coefficients*). For each problem, we aim to minimize the loss function in Eq. 3. We use a 4-layer fully-connected NN with 50 neurons per layer, a hyperbolic tangent activation function, and randomly sample collocation points (x, t) on the domain. Furthermore, all the systems that we consider have periodic boundary conditions. We enforce this through an extra term in the loss function that takes the difference between the predicted NN solution at each boundary. We train this network using the L-BFGS optimizer and sweep over learning rates from $1e-4$ to 2.0 .² After training the PINN, we measure the L_2 relative and absolute errors between the PINN’s predicted solution and the analytical solution. The L_2 relative error is $\frac{1}{N} \sum_{i=0}^N \frac{\|\hat{u} - u\|_2}{\|u\|_2}$; and the absolute error is $\frac{1}{N} \sum_{i=0}^N \|\hat{u} - u\|_2$, where N is the number of evaluation grid points, \hat{u} is the predicted solution by the PINN, and u is the true solution. For all cases, we run models at least ten times with different preset random seeds, and we average the relative and absolute errors in $u(x, t)$. For each loss function, \hat{u} is the output of the NN and shorthand for $\hat{u} = NN(\theta, x, t)$.

3.1 Learning convection

Problem formulation. We first consider a one-dimensional convection problem, a hyperbolic PDE which is commonly used to model transport phenomena:

$$\begin{aligned} \frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} &= 0, \quad x \in \Omega, \quad t \in [0, T], \\ u(x, 0) &= h(x), \quad x \in \Omega. \end{aligned} \tag{5}$$

¹Note that for convection, this does not necessarily mean that the mapping from initial to final solution is also linear. This just means that the terms in the PDE are linear.

²For reasons that are only partially understood, L-BFGS methods tend to perform better for existing PINN problems. While variants of stochastic gradient descent are much more popular in computer vision, natural language processing, and recommendation systems, we found that they underperform in comparison to L-BFGS and as such we keep this setting to be the same as in the original PINN paper [14].

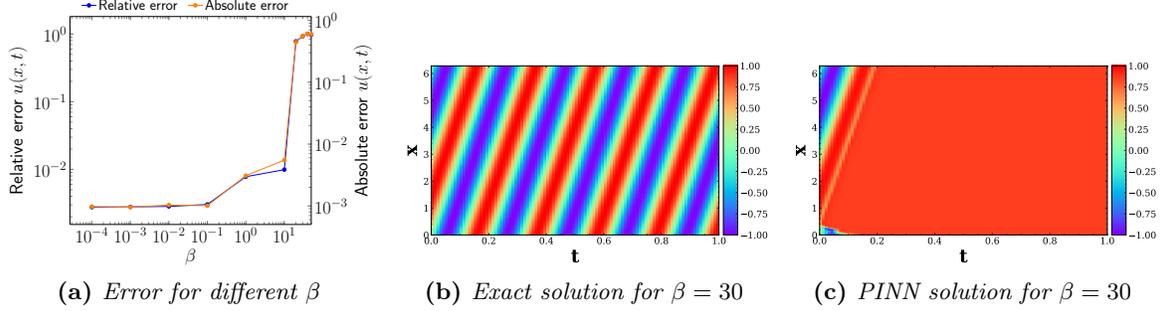


Figure 1: Prediction error for 1D convection (§3.1) problem, when β is changed. The PINN has difficulty predicting the solution past a certain timestep, but is able to fit the boundary conditions. Additional figures for different β values can be seen in Fig. B.1.

Here, β is the convection coefficient and $h(x)$ is the initial condition. For constant β and periodic boundary conditions, this problem has a simple analytical solution:

$$u_{\text{analytical}}(x, t) = F^{-1}(F(h(x))e^{-i\beta kt}), \quad (6)$$

where F is the Fourier transform, $i = \sqrt{-1}$, and k denotes frequency in the Fourier domain. The general loss function for this problem (corresponding to Eq. 3) is

$$\mathcal{L}(\theta) = \frac{1}{N_u} \sum_{i=1}^{N_u} (\hat{u} - u_0^i)^2 + \frac{1}{N_f} \sum_{i=1}^{N_f} \lambda_i \left(\frac{\partial \hat{u}}{\partial t} + \beta \frac{\partial \hat{u}}{\partial x} \right)^2 + \mathcal{L}_B, \quad (7)$$

where $\hat{u} = NN(\theta, x, t)$ is the output of the NN, and \mathcal{L}_B is the boundary loss. For periodic boundary conditions with $\Omega = [0, 2\pi)$, this loss is:

$$\mathcal{L}_B = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(\hat{u}(\theta, 0, t) - \hat{u}(\theta, 2\pi, t) \right)^2. \quad (8)$$

We use the following simple initial and periodic boundary conditions:

$$\begin{aligned} u(x, 0) &= \sin(x), \\ u(0, t) &= u(2\pi, t). \end{aligned} \quad (9)$$

Observations. We apply the PINN’s soft regularization to this problem, and we optimize the loss function in Eq. 7. After training, we measure the relative and absolute errors between the PINN’s predicted solution and the analytical solution, as reported in Fig. 1(a). As one can see, the PINN is only able to achieve good solutions for small values of convection coefficient, and it fails when β becomes larger, reaching a relative error of almost 100% for $\beta > 10$. We also provide visualization of the exact and PINN solution in Fig. 1(b-c). One can clearly see that the PINN solution is unable to learn the solution. As we will later show, the NN architecture does have enough capacity to find the solution, but that the training/optimization problem is very difficult to solve with PINNs (and importantly, we were not able to improve the results even after extensive hyperparameter tuning).

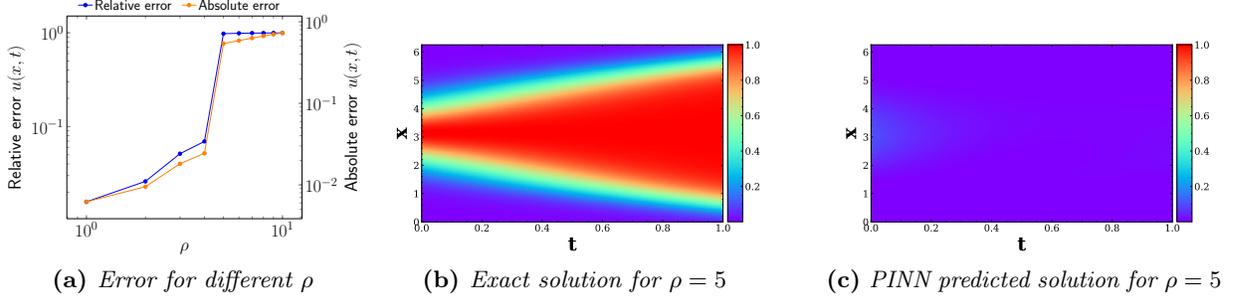


Figure 2: Prediction error for 1D reaction (§3.2) problem, when ρ is changed. The PINN has difficulty predicting any part of the solution (and can't even predict the first few timesteps), and only predicts a homogeneous solution everywhere.

3.2 Learning reaction

Problem formulation. Here we consider an example of a one-dimensional reaction equation, which is commonly used to model chemical reactions. We look at the reaction term in Fisher's equation, which is a semi-linear ordinary differential equation:

$$\begin{aligned} \frac{\partial u}{\partial t} - \rho u(1 - u) &= 0, \quad x \in \Omega, t \in (0, T], \\ u(x, 0) &= h(x), \quad x \in \Omega. \end{aligned} \quad (10)$$

Here, ρ is the reaction coefficient and $h(x)$ is the initial condition. The reaction problem has a simple analytical solution for periodic boundary conditions and constant ρ :

$$u_{\text{analytical}}(x, t) = \frac{h(x)e^{\rho t}}{h(x)e^{\rho t} + 1 - h(x)}. \quad (11)$$

The general loss function for this problem is

$$\mathcal{L}(\theta) = \frac{1}{N_u} \sum_{i=1}^{N_u} (\hat{u} - u_0^i)^2 + \frac{1}{N_f} \sum_{i=1}^{N_f} \lambda_i \left(\frac{\partial \hat{u}}{\partial t} - \rho \hat{u}(1 - \hat{u}) \right)^2 + \mathcal{L}_B, \quad (12)$$

where \mathcal{L}_B is the boundary loss (same as in Eq. 8). We consider the following initial and boundary conditions:

$$\begin{aligned} u(x, 0) &= e^{-\frac{(x-\pi)^2}{2(\pi/4)^2}}, \\ u(0, t) &= u(2\pi, t). \end{aligned} \quad (13)$$

Observations. We report the relative/absolute error of the PINN with respect to the ground truth in Fig. 2. Similar to the convection case, we can see that the PINN is only able to learn the problem for very small values of the reaction coefficient, ρ . However, the error quickly gets to 100% as ρ is increased. The example heatmap in Fig. 2(c) shows that the PINN is not able to predict the solution at all, and instead it predicts a mostly homogeneous solution, close to zero, everywhere.

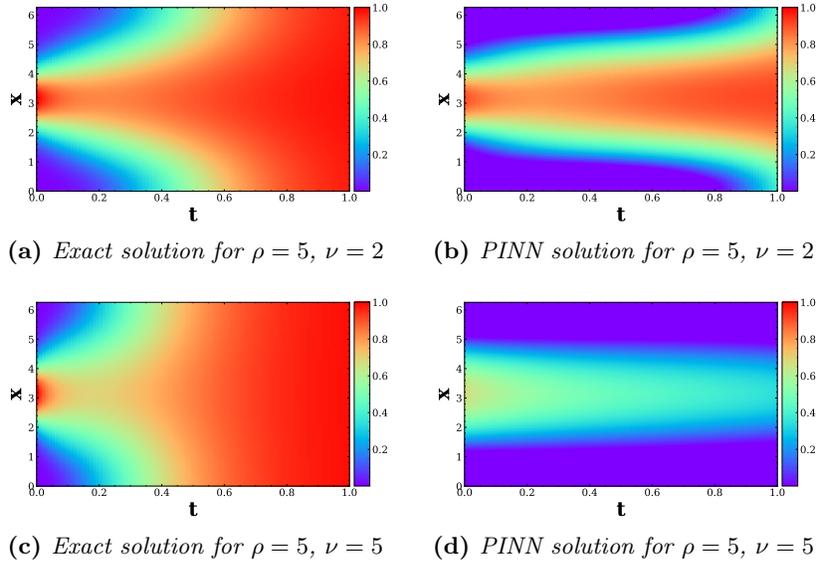


Figure 3: Prediction error for 1D reaction-diffusion (§3.3) problem. We can clearly see that the PINN has difficulty predicting the solution (especially the “sharpness” of the solution) and is unable to capture the correct behavior. Additional figures for different ν values can be seen in Fig. C.1.

3.3 Learning reaction-diffusion

Problem formulation. We next look at a reaction-diffusion system, where we add a diffusion operator to the reaction equation discussed above. Note that for pure diffusion, the solution dissipates to a steady-state of uniform/constant distribution which may be trivial to learn. Therefore, we consider studying the reaction-diffusion system:

$$\begin{aligned} \frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1 - u) &= 0, \quad x \in \Omega, \quad t \in (0, T], \\ u(x, 0) &= h(x), \quad x \in \Omega. \end{aligned} \quad (14)$$

Here, ν ($\nu > 0$) is the diffusion coefficient. The solution of such a system can be solved for via Strang splitting, i.e., splitting the equation into two separate models (a reaction component and a diffusion component):

$$\begin{aligned} \frac{du}{dt} &= \rho u(1 - u) \\ \frac{du}{dt} &= \nu \frac{\partial^2 u}{\partial x^2} \end{aligned} \quad (15)$$

For each timestep, we can solve the reaction equation through Eq. 11. The diffusion equation has the following analytical solution:

$$u_{\text{analytical}}(x, t) = F^{-1}(F(u(x, t = t^n))e^{-\nu k^2 t}), \quad (16)$$

where $u(x, t = t^n)$ is the solution at the n^{th} time step. We solve the reaction equation for each timestep, and then use the reaction solution as the initial condition to solve the diffusion component and get the final solution.

The general loss function for this problem is,

$$\mathcal{L}(\theta) = \frac{1}{N_u} \sum_{i=1}^{N_u} (\hat{u} - u_0^i)^2 + \frac{1}{N_f} \sum_{i=1}^{N_f} \lambda_i \left(\frac{\partial \hat{u}}{\partial t} - \nu \frac{\partial^2 \hat{u}}{\partial x^2} - \rho \hat{u}(1 - \hat{u}) \right)^2 + \mathcal{L}_{\mathcal{B}}, \quad (17)$$

where $\mathcal{L}_{\mathcal{B}}$ is the boundary loss. Similar to the previous examples, periodic boundary conditions can be enforced by including $\mathcal{L}_{\mathcal{B}}$ from Eq. 8 as an extra term in the loss.

Observations. Similar to the previous cases, we can see that the PINN also fails to learn reaction-diffusion. We illustrate two cases in Fig. 3 with $\rho = 5$, when $\nu = 2$ and $\nu = 5$. In particular, for $\nu = 2$ the PINN achieves a relative error of 50%. We can see that it is unable to capture the “sharper” transitions, though it can predict the center of the solution a little better. Similarly, for $\nu = 5$ the PINN achieves a higher relative error of 93%. Here, we can clearly see that the PINN is unable to capture either the reaction or diffusion component.

4 Diagnosing possible failure modes for physics-informed NNs

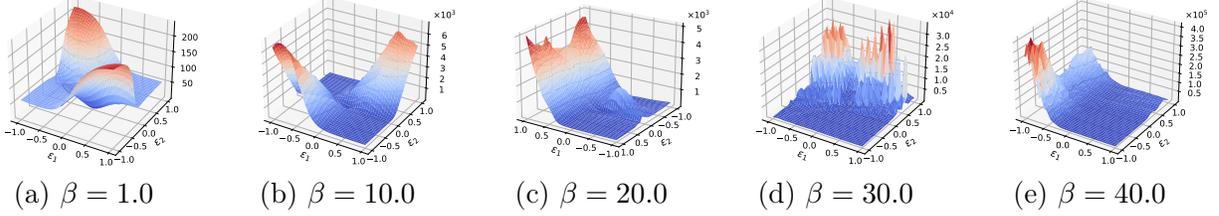
Thus far, we have shown that PINNs can result in high errors even for simple physical regimes, in particular for PDEs/ODEs with non-trivial convection/reaction/diffusion coefficients. Here, we demonstrate that one of the underlying reasons for this is introduced through the PDE-based soft constraint of $\mathcal{L}_{\mathcal{F}}$, which makes the loss landscape difficult to optimize. We first (in §4.1) analyze the loss landscape to illustrate how increasing this soft regularization can lead to more complex loss landscapes, thus leading to optimization difficulties. We then (in §4.2) demonstrate how this is related to regularizing with differential operators, which can result in ill-conditioning.

4.1 Soft PDE regularization and optimization difficulties

Here, we analyze how the loss landscape changes for different regimes for the convection problem in §3.1 with/without the soft regularization in PINNs. We show that adding the soft regularization can actually make the problem harder to optimize, i.e., the regularization leads to less smooth loss landscapes. For all the experiments, we plot the loss landscape by perturbing the (trained) model across the first two dominant Hessian eigenvectors and computing the corresponding loss values. This tends to be more informative than perturbing the model parameters in random directions [36, 37].

Figure 4 shows the loss landscape for the convection problem (discussed in §3.1), for different β values. Interestingly, the loss landscape at a relatively low $\beta = 1$ is rather smooth, but increasing β further results in a complex and non-symmetric loss landscape. It is also evident that the optimizer has gotten stuck in a local minima with a very high loss function for large β values.

Finally, we study the impact of changing the weight/multiplier for the soft regularization term (i.e., the λ parameter in Eq. 3), which has been argued to be important in improving PINN performance [27]. While we find that tuning λ can help change the error, it cannot resolve the problem, as shown in Fig. 5. Note that as the regularization parameter is increased, the loss landscape becomes increasingly more complex and harder to optimize (additionally, see the z-axis scale).

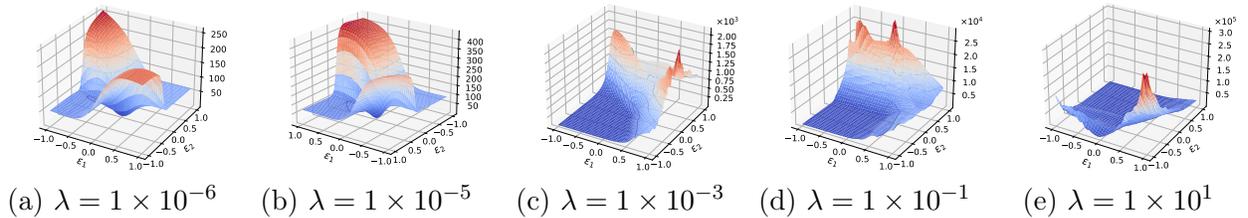


β	1	10	20	30	40
Relative error	7.84×10^{-3}	1.08×10^{-2}	7.50×10^{-1}	8.97×10^{-1}	9.61×10^{-1}
Absolute error	3.17×10^{-3}	6.03×10^{-3}	4.32×10^{-1}	5.42×10^{-1}	5.82×10^{-1}

Figure 4: Loss landscapes for varying values of β , for the 1D convection example in §3.1. The loss landscape is more smooth at low β , and it becomes increasingly more complex as β increases, which can make the optimization problem more difficult. In particular, at higher β , the optimizer gets stuck in a flat regime. These results support that adding the PDE soft regularization term results in a more complex optimization loss landscape.

4.2 A PDE perspective on ill-conditioned regularization

One of the difficulties with PINNs arises from the soft regularization term that includes differential operators. This term is quite different from norm-based regularization that is more common in ML, and this can actually make the problem more ill-conditioned (or less regularized). From a PDE perspective, this is not surprising as the PDE-based regularization operator (i.e., $\mathcal{L}_{\mathcal{F}}$ term) in the PINN can in fact be ill-conditioned, which can lead to unstable numerical behavior. For example, in numerical PDE analysis, it is well-known that the condition number for the regularization operator for the diffusion problem is $\mathcal{O}(\nu N^2)^2$, where N is the grid size (see §A for the derivation). Similarly, the condition number for the convection problem scales as $\mathcal{O}(\beta N)^2$, which is still quite high. As such, it is not surprising that this ill-conditioned property would lead to instability which can manifest



λ	1×10^{-6}	1×10^{-5}	1×10^{-3}	1×10^{-1}	1×10^1
Relative error	1.69	1.65	1.00	1.08	0.982
Absolute error	0.987	0.987	0.623	0.647	0.595

Figure 5: Loss landscapes when varying the λ parameter in \mathcal{F} , for the 1D convection equation in §3.1. In this example, $\beta = 30$, which is a point at which the error is high. The loss landscape becomes more complex as λ is increased, i.e., as the regularization term grows. However, error stays consistently high (although it decreases a little as λ is increased).

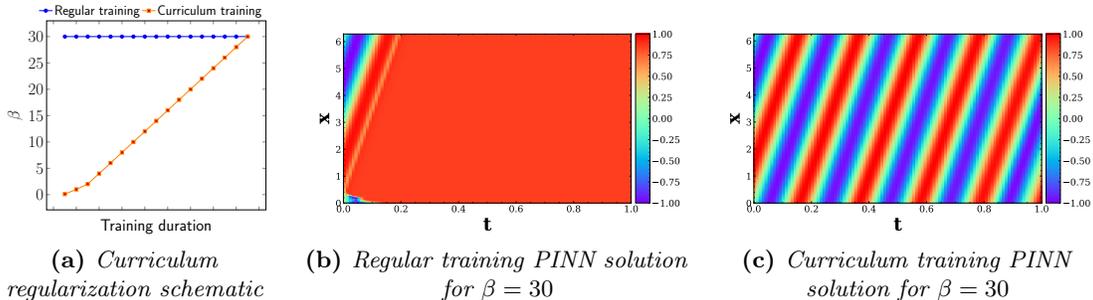


Figure 6: Schematic outlining curriculum regularization and example result for 1D convection from §3.1 The training procedure for regular PINNs training versus curriculum PINN training for the convection example in §3.1. The regular PINN training only involves training at $\beta = 30$, while curriculum regularization starts at a lower β , trains a model, and then uses the weights of this model to reinitialize the NN for training the next β . The curriculum training approach is able to do significantly better (by almost two orders of magnitude).

itself in large gradients and/or poor convergence behavior. (Similar results were also reported in [27].) We should however emphasize that the condition number is only one of the many factors involved in this problem, and other factors such as the complexity of the function that we are trying to approximate, the non-convex loss landscape, and the limitations of optimization algorithms need to be considered. To give an example, a highly diffusive PDE with a very large diffusion coefficient quickly results in a solution that approaches uniform/constant distribution, which can be very easy to approximate with trivial initial conditions. However, as with other ill-conditioned operators, it is in the presence of noise, or cases with non-trivial physics, that the instability appears.

5 Expressivity versus optimization difficulty

In this section, we first show that the failure modes we observed are not necessarily due to the specific NN architecture that we used in our experiments. In particular, we show that the NN model does have the expressivity/capacity to learn the convection/reaction/diffusion coefficient cases where the vanilla PINN method fails. Additionally, in the process of demonstrating this, we also describe two methods that lead to significantly lower error rates. In particular, we show that changing the learning paradigm to *curriculum regularization* can make the optimization problem easier to solve (as discussed in §5.1). Second, we show that posing the problem as *sequence-to-sequence learning* may lead to better results than learning the entire state-space at once (as discussed in §5.2).

5.1 Curriculum PINN Regularization

One may contend that the failure modes shown in §3 may be because the NN does not have enough capacity. Here, we show that this is not the underlying reason. To do so, we devise a “curriculum regularization” method to warm start the NN training by finding a good initialization for the weights. Instead of training the PINN to learn the solution right away for cases with higher β/ρ , we start by training the PINN on lower β/ρ (easier for the PINN to learn) and then gradually move to training the PINN on higher β/ρ , respectively. We test these results for the examples in §3.1 and §3.2. This is somewhat analogous to curriculum learning in ML [38], but applied by progressively making the

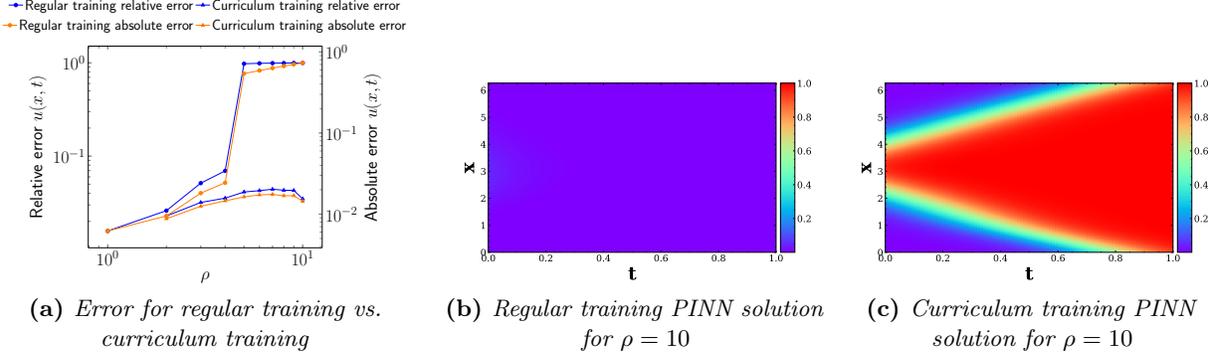


Figure 7: Curriculum regularization for 1D reaction from §3.2. The PINN is now able to predict the solution much more closely, including capturing the “sharp” features (traditionally hard for PINNs), and error is 1-2 orders of magnitude lower when using curriculum training over regular training.

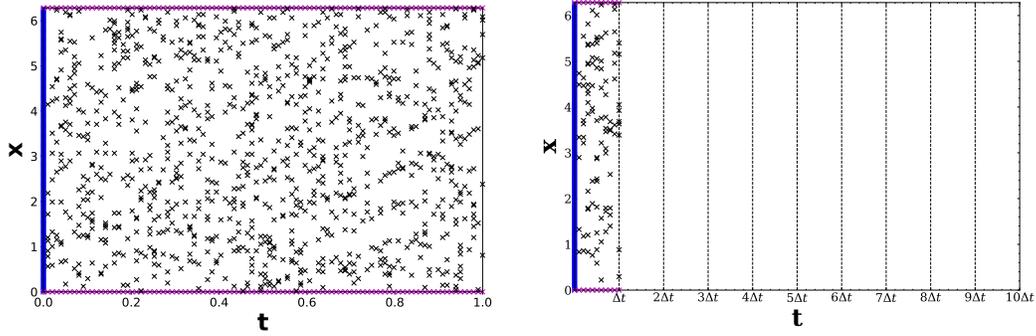
		Regular PINN	Curriculum training
1D convection: $\beta = 20$	Relative error	7.50×10^{-1}	9.84×10^{-3}
	Absolute error	4.32×10^{-1}	5.42×10^{-3}
1D convection: $\beta = 30$	Relative error	8.97×10^{-1}	2.02×10^{-2}
	Absolute error	5.42×10^{-1}	1.10×10^{-2}
1D convection: $\beta = 40$	Relative error	9.61×10^{-1}	5.33×10^{-2}
	Absolute error	5.82×10^{-1}	2.69×10^{-2}

Table 1: Training the PINN gradually on more difficult problems improves performance. 1D convection example in §3.1. The curriculum training approach achieves significantly better errors.

PDE/ODE harder to solve.

Figure 6 shows the training procedure for an example convection case (§3.1) with $\beta = 30$. As Fig. 6(c) shows, the curriculum regularization approach results in a much more accurate solution than regular PINN training. With curriculum regularization, the relative error is almost two orders of magnitude lower. Additionally, this is true across all the other regimes that we found regular PINNs to fail, as shown in Tab. 1. In Fig. D.1, we also show that curriculum regularization not only decreases error significantly, but also decreases the variance of the error.

Curriculum regularization also works well for the reaction example in §3.2. In this case, we start by training with a low ρ value (reaction coefficient), and then increase gradually to higher ρ values. The results can be seen in Fig. 7. We can see that the error is 0.1 - 0.6 orders of magnitude lower for $\rho = 2 - 4$ (when the regular PINN error is not as high), and then greatly decreases error by 1-2 orders of magnitude for $\rho = 5 - 10$. As we discussed before, PINN has difficulty in learning sharp features for high values of ρ . However, the curriculum regularization overcomes this, even for $\rho = 10$, as seen in Fig. 7(c).



(a) Regular PINN training

(b) Sequence-to-sequence learning (model trained every Δt)

× Initial condition points × Boundary points × Collocation points

Figure 8: Schematic outlining seq2seq learning. In contrast to regular PINN training, the solution in seq2seq learning is predicted for only one Δt step at a time. Then, the predicted solution at $t = \Delta t$ is used as the initial condition for the next segment. To allow fair comparison, we keep the total number of collocation points to be exactly the same in either approach. That is, we do not increase the number of collocation points for seq2seq learning in the right, and keep it to be the same as in the corresponding segment in the left figure.

		Entire state space	$\Delta t = 0.05$	$\Delta t = 0.1$
$\rho = 5$	Relative error	9.79×10^{-1}	7.06×10^{-2}	7.09×10^{-2}
	Absolute error	5.40×10^{-1}	2.52×10^{-2}	2.39×10^{-2}
$\rho = 6$	Relative error	9.88×10^{-1}	8.25×10^{-2}	7.78×10^{-2}
	Absolute error	5.88×10^{-1}	3.02×10^{-2}	2.65×10^{-2}
$\rho = 7$	Relative error	9.92×10^{-1}	8.16×10^{-2}	7.56×10^{-2}
	Absolute error	6.31×10^{-1}	3.03×10^{-2}	2.69×10^{-2}
$\rho = 8$	Relative error	9.94×10^{-1}	8.19×10^{-2}	7.44×10^{-2}
	Absolute error	6.69×10^{-1}	3.10×10^{-2}	2.73×10^{-2}
$\rho = 9$	Relative error	9.95×10^{-1}	7.02×10^{-2}	8.63×10^{-2}
	Absolute error	7.02×10^{-1}	2.83×10^{-2}	3.21×10^{-2}
$\rho = 10$	Relative error	9.96×10^{-1}	6.88×10^{-2}	7.47×10^{-2}
	Absolute error	7.31×10^{-1}	2.85×10^{-2}	2.85×10^{-2}

Table 2: Predicting the entire state space versus discretizing the state space (i.e., seq2seq learning) for 1D reaction (§3.2). The seq2seq learning achieves lower error for both $\Delta t = 0.05$ and $\Delta t = 0.1$, in comparison to the PINN’s approach of predicting the entire state space at once.

5.2 Sequence-to-sequence learning vs learning the entire space-time solution

The original PINN approach of [14] trains the NN model to predict the entire space-time at once (i.e., predict u for all locations and time points). In certain cases, this can be more difficult to learn.

		Entire state space	$\Delta t = 0.05$	$\Delta t = 0.1$
$\nu = 2, \rho = 5$	Relative error	5.07×10^{-1}	2.04×10^{-2}	1.18×10^{-2}
	Absolute error	2.70×10^{-1}	1.06×10^{-2}	6.41×10^{-3}
$\nu = 3, \rho = 5$	Relative error	7.98×10^{-1}	1.92×10^{-2}	1.56×10^{-2}
	Absolute error	4.79×10^{-1}	1.01×10^{-2}	8.17×10^{-3}
$\nu = 4, \rho = 5$	Relative error	8.84×10^{-1}	2.37×10^{-2}	1.59×10^{-2}
	Absolute error	5.74×10^{-1}	1.15×10^{-2}	8.01×10^{-3}
$\nu = 5, \rho = 5$	Relative error	9.35×10^{-1}	2.36×10^{-2}	2.39×10^{-2}
	Absolute error	6.46×10^{-1}	1.09×10^{-2}	1.15×10^{-2}
$\nu = 6, \rho = 5$	Relative error	9.60×10^{-1}	2.81×10^{-2}	2.69×10^{-2}
	Absolute error	6.84×10^{-1}	1.17×10^{-2}	1.28×10^{-2}

Table 3: *Predicting the entire state space versus discretizing the state space (i.e., seq2seq learning) for 1D reaction-diffusion (§3.3). The seq2seq learning achieves lower error for both $\Delta t = 0.05$ and $\Delta t = 0.1$, in comparison to the PINN’s approach of predicting the entire state space at once.*

Here, we demonstrate that it may be better to pose the problem as a sequence-to-sequence (seq2seq) learning task, where the NN learns to predict the solution at the next time step, instead of all times. This way, we can use a marching-in-time scheme to predict different sequences/time points. Note that the only data available here is from the PDE itself, i.e., just the initial condition. We take the prediction at $t = \Delta t$ and use this as the initial condition to make a prediction at $t = 2\Delta t$, and so on. This is schematically outlined in Fig. 8.

We test this scheme by using the exact same NN architecture as in previous sections, and we report the results in Tab. D.1 for the convection problem of §3.1, Tab. 2 for the reaction problem of §3.2, and Tab. 3 for the reaction-diffusion problem of §3.3. We compare the relative/absolute error when the learning is posed as a seq2seq problem (i.e., predicting the state space with a “time marching scheme” of one timestep prediction at a time) to the PINN approach of predicting the whole state space at once.³

We explore the following cases where the PINN does poorly, varying β , ρ , and ν coefficients:

- 1) For 1D convection (§3.1), higher β values from 30-40.
- 2) For 1D reaction (§3.2), ρ coefficients from 5-10.
- 3) For 1D reaction-diffusion (§3.3), a fixed $\rho = 5$ and ν coefficients from 2-6.

For these cases, we find that posing the problem as seq2seq learning results in significantly lower error. The difference is particularly striking for the reaction and reaction-diffusion cases, where the seq2seq PINN model decreases error by almost two orders of magnitude. An example case is shown Fig. 9, where the seq2seq approach is able to recover the solution, while regular PINNs does very poorly. Note that this behavior also has analogues with numerical methods used in scientific computing, where space-time problems are typically harder to solve for, as compared

³To have a fair comparison between the two methods (time marching versus predicting entire state space at once), for the time marching method, we use the same number of collocation (interior) points for both. For example, for $T = [0, 1]$, if we use 1000 collocation points to predict the entire state space, then for $\Delta t = 0.1$ we use 100 collocation points per section.

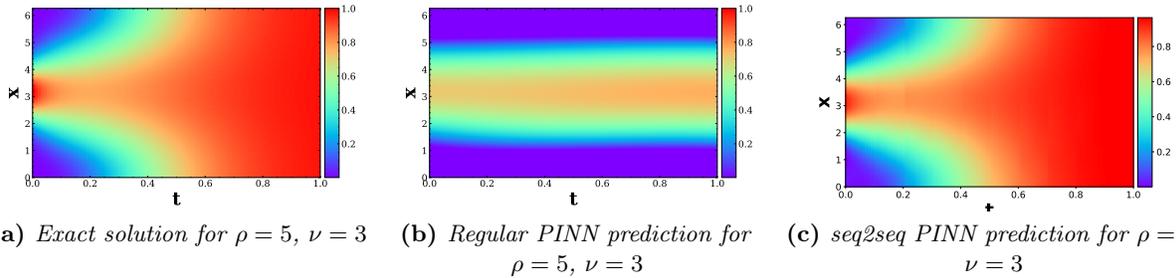


Figure 9: Predicting the entire state space vs seq2seq learning for 1D reaction-diffusion. The regular PINN is unable to capture the “sharp” and/or diffusive features correctly. However, the seq2seq learning approach is able to capture the correct solution, and achieves almost two orders of magnitude lower error.

to time marching methods [39]. Intuitively, since the problem is ill-conditioned, restricting the dimensions is expected to help. Furthermore, the underlying function/mapping of the input to the solution should be much simpler to approximate over a smaller time span, as compared to the full time horizon.

These initial results are promising, and further developments may lead to still better ways of using PINNs and learning PDEs. In particular, using more sophisticated methods to predict timesteps across the state space may provide improved performance, as may including more sophisticated seq2seq approaches and tuning the regularization parameter (i.e., amount of constraint added).

6 Conclusions

PINNs—and SciML more generally—hold great promise for expanding the scope of ML methodology to important problems in science and engineering. For these problems, however, integrating data-driven ML methods with domain-driven PDE-based constraints as a soft regularization term can lead to subtle problems which are not common in classical ML problems. In particular, we showed that this approach can have fundamental limitations which results in failure modes for learning relevant physics commonly used in different fields of science. To show this, we picked three fundamental PDE/ODE problems: convection, reaction, and reaction-diffusion. We showed that the basic PINN approach only works for very simple cases, failing to learn the relevant physical phenomena for moderately more challenging regimes—including physically-relevant non-trivial values of physical parameters. Our analysis of the underlying failure reasons showed that adding the PINN soft regularization makes the loss landscape increasingly more complex (even for moderate values of convection, reaction, or diffusion coefficients). We demonstrated that one of the underlying problems relates to the ill-conditioned nature of PDEs. We then proposed different methods to address these issues, with promising results. This includes posing the PDE/ODE problem in a way similar to curriculum learning. Furthermore, we showed that using a sequence-to-sequence learning procedure (rather than learning to predict the entire space-time at once) can improve training, especially for the reaction and reaction-diffusion problem. We have open sourced our PyTorch based framework [8] both to help with reproducibility and to enable other researchers to extend the results.

Acknowledgements

The authors are especially thankful to Shashank Subramanian for his feedback and contributions. We also acknowledge helpful discussions with Prof. George Biros, Geoffrey Negiar, and Daniel Rothchild. Aditi S. Krishnapriyan was supported by Laboratory Directed Research and Development (LDRD) funding under Contract Number DE-AC02-05CH11231 at LBNL and the Alvarez Fellowship in the Computational Research Division at LBNL. Amir Gholami was supported through funding from Samsung SAIT. Michael W. Mahoney would also like to acknowledge the UC Berkeley CLTC, ARO, NSF, and ONR. The UC Berkeley team also acknowledges gracious support from Intel corporation, Intel VLAB, Samsung, Amazon AWS, Google Cloud, Google TPU Research Cloud, and Google Brain (in particular Prof. David Patterson, Dr. Ed Chi, and Jing Li). Our conclusions do not necessarily reflect the position or the policy of our sponsors, and no official endorsement should be inferred.

References

- [1] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Perumal Nithiarasu, and JZ Zhu. *The finite element method*, volume 3. McGraw-hill London, 1977.
- [2] Bengt Fornberg. *A practical guide to pseudospectral methods*. Number 1. Cambridge university press, 1998.
- [3] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 2021.
- [4] Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. Informed machine learning—a taxonomy and survey of integrating knowledge into learning systems. *arXiv preprint arXiv:1903.12394*, 2019.
- [5] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 2020.
- [6] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.
- [7] Parviz Moin. *Fundamentals of engineering numerical analysis*. Cambridge University Press, 2010.
- [8] Characterizing possible failure modes in physics-informed neural networks. <https://github.com/a1k12/characterizing-pinns-failure-modes>, 2021.
- [9] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216. PMLR, 2018.
- [10] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

- [11] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.
- [12] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.
- [13] E Weinan, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.
- [14] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [15] Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Zhiwei Fang, Max Rietmann, Wonmin Byeon, and Sanjay Choudhry. Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In *International Conference on Computational Science*, pages 447–461. Springer, 2021.
- [16] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.
- [17] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8):11618–11633, 2020.
- [18] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.
- [19] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.
- [20] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.
- [21] Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.
- [22] Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E Hurtado, and Ellen Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020.
- [23] Weiqi Ji, Weilun Qiu, Zhiyu Shi, Shaowu Pan, and Sili Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *arXiv preprint arXiv:2011.04520*, 2020.

- [24] Olga Fuks and Hamdi A Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020.
- [25] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *arXiv preprint arXiv:2007.14527*, 2020.
- [26] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *arXiv preprint arXiv:2012.10047*, 2020.
- [27] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*, 2020.
- [28] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32:15379–15389, 2019.
- [29] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.
- [30] Bo Wang, Wenzhong Zhang, and Wei Cai. Multi-scale deep neural network (mscalednn) methods for oscillatory stokes flows in complex domains. *arXiv preprint arXiv:2009.12729*, 2020.
- [31] Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep networks: Promises and limitations. *arXiv preprint arXiv:1706.02025*, 2017.
- [32] Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.
- [33] Kailai Xu and Eric Darve. Physics constrained learning for data-driven inverse modeling from sparse observations. *arXiv preprint arXiv:2002.10521*, 2020.
- [34] Yatin Nandwani, Abhishek Pathak, Parag Singla, et al. A primal dual formulation for deep learning with constraints. *Advances in Neural Information Processing Systems*, 2019.
- [35] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *arXiv preprint arXiv:2102.04626*, 2021.
- [36] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. *Advances in Neural Information Processing Systems*, 2018.
- [37] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 581–590. IEEE, 2020.
- [38] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
- [39] Kenneth Eriksson, Don Estep, Peter Hansbo, and Claes Johnson. *Computational differential equations*. Cambridge University Press, 1996.

A Approximate condition number scaling for PDE regularization in PINNs

We provide a derivation to obtain an approximate condition number for the PINN regularization term. First, note that the condition number of an operator is a metric for the amount that a function's output changes for a change in its input. Using this definition, we can obtain an approximate bound on the condition number for the PINN regularization operator. We should emphasize that the non-linear nature of the regularization makes the exact condition number behavior dependent on the state variable and its derivatives, and so the results obtained below are approximate.

Convection problem. For the convection problem, the PDE-based regularization operator is:

$$f = \frac{du}{dt} + \beta \frac{\partial u}{\partial x}. \quad (18)$$

For a small change in δu to input, the output will change as follows:

$$\delta f = \frac{d\delta u}{dt} + \beta \frac{\partial \delta u}{\partial x}. \quad (19)$$

We can estimate that the maximum change in output (denoted as δf) is proportional to the sum of the maximum change in the first time derivative, which scales as $\mathcal{O}(\delta t^{-1})$, and the second term, which scales as $\mathcal{O}(\beta h^{-1})$. Here, δt is the time step size and $h = 1/N$ is the grid size spacing for a uniformly discretized grid of size N . Assuming that the time step size is not very small, the condition number will be proportional to the second term and will scale as $\mathcal{O}(\beta N)$. Since the PINN regularization term uses L_2 loss, this change will be quadratically scaled.

Diffusion problem. Similarly, we can obtain an approximate scaling of the condition number for the diffusion case where the regularization function has the following form:

$$f = \frac{du}{dt} - \nu \frac{\partial^2 u}{\partial x^2}. \quad (20)$$

The corresponding change in the output, for a perturbation in u , is then:

$$\delta f = \frac{d\delta u}{dt} - \nu \frac{\partial^2 \delta u}{\partial x^2}. \quad (21)$$

Similar to the previous case, we can estimate that the maximum change in output (denoted as δf) is proportional to the sum of the maximum change in the first time, which again scales as $\mathcal{O}(\delta t^{-1})$, and the second term, which scales as $\mathcal{O}(\nu h^{-2})$. Again, assuming that the time step size is not very small, the condition number will be proportional to the second term and will scale as $\mathcal{O}(\nu N^2)$. Similarly, since the PINN regularization term uses L_2 loss, this change will be quadratically scaled.

Also, we observed that error increases more rapidly when ν increases (for the diffusion case), in contrast to the slower error increase with respect to β for the convection case. Finally, as our estimates of the condition number show, the condition number for diffusion scales N^2 while convection scales N .

B Learning convection

We include additional heatmaps for learning convection (§3.1).

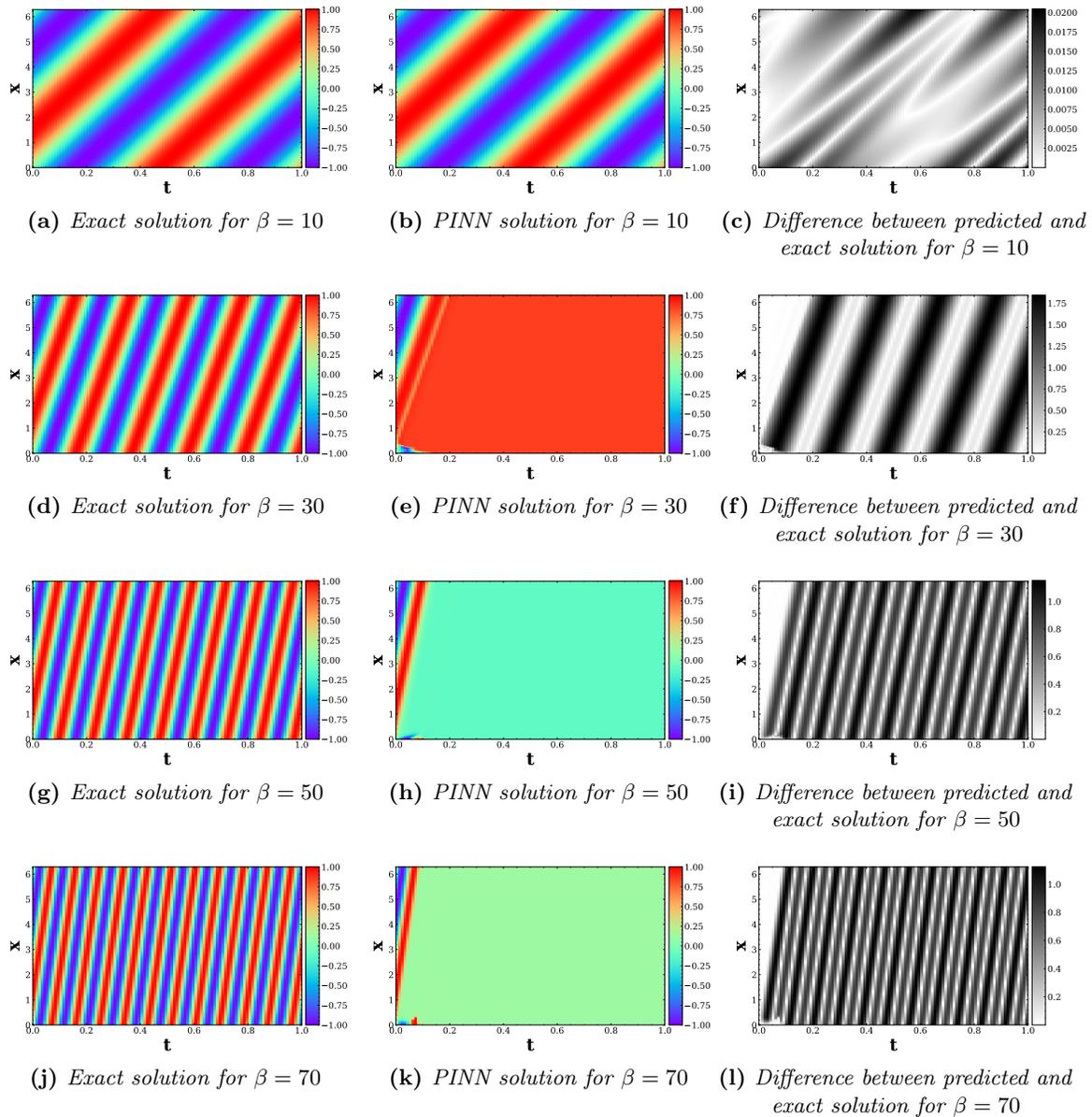


Figure B.1: Heatmap of exact vs predicted solution for 1D convection (§3.1). Heatmap of the exact solutions to the 1D convection equation, Eq. 5, for a variety of β values and the respective PINN predicted solution. The PINN is unable to predict the solution as β increases, past a certain timestep.

C Learning reaction-diffusion

We include additional heatmaps for learning reaction-diffusion (§3.3).

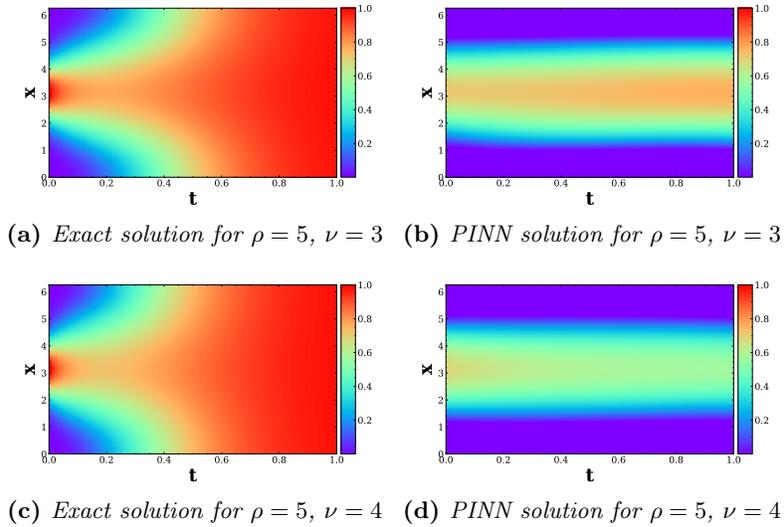


Figure C.1: Heatmap of exact vs predicted solution for 1D reaction-diffusion (§3.3). Heatmap of the exact solutions to the 1D reaction-diffusion equation, Eq. 14, for different ν values and the respective physics-informed NN predicted solution. The PINN is unable to predict the solution, including both capturing the “sharp” features and/or diffuse features.

D Extra Results

D.1 Extra curriculum regularization results

We include extra curriculum regularization results for §3.1. These results demonstrate that curriculum regularization not only decreases the error in the solution, but also decreases the variance in the error.

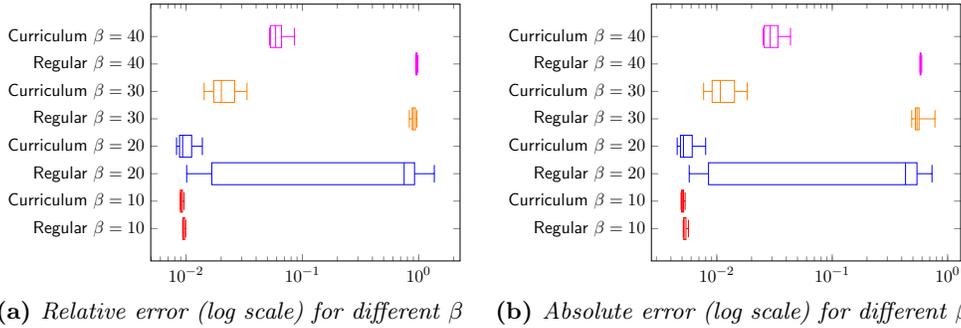


Figure D.1: Training the PINN gradually on more difficult problems improves performance. 1D convection example in §3.1. Summary of performance across 10 preset random seeds (with lowest error per seed) to show the variance in error. The curriculum learning approach achieves significantly better errors, as well as lower variance in the error.

D.2 Extra sequence-to-sequence learning results

We include sequence-to-sequence learning results for convection (§3.1).

		Entire state space	$\Delta t = 0.05$	$\Delta t = 0.1$
$\beta = 30$	Relative error	7.38×10^{-1}	2.13×10^{-1}	1.05×10^{-1}
	Absolute error	5.57×10^{-1}	1.29×10^{-1}	5.95×10^{-2}
$\beta = 40$	Relative error	8.25×10^{-1}	4.58×10^{-1}	2.41×10^{-1}
	Absolute error	6.06×10^{-1}	2.58×10^{-1}	1.35×10^{-1}

Table D.1: Predicting the entire state space versus discretizing the state space (i.e., seq2seq learning) for 1D convection (§3.1). The seq2seq learning achieves lower error for both $\Delta t = 0.05$ and $\Delta t = 0.1$, in comparison to the PINN’s approach of predicting the entire state space at once. For this example only, we use a higher number of collocation points (10000) for regular PINNs and seq2seq learning, which minimizes the variance in the seq2seq results.