Multifidelity Modeling for Physics-Informed Neural Networks (PINNs)

Michael Penwarden[a], Shandian Zhe[b], Akil Narayan[c], Robert M. Kirby[a]

[a]*School of Computing and Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT*
[b]*School of Computing, University of Utah, Salt Lake City, UT*
[c]*Department of Mathematics and Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT*

Abstract

Multifidelity simulation methodologies are often used in an attempt to judiciously combine low-fidelity and high-fidelity simulation results in an accuracy-increasing, cost-saving way. Candidates for this approach are simulation methodologies for which there are fidelity differences connected with significant computational cost differences. Physics-informed Neural Networks (PINNs) are candidates for these types of approaches due to the significant difference in training times required when different fidelities (expressed in terms of architecture width and depth as well as optimization criteria) are employed. In this paper, we propose a particular multifidelity approach applied to PINNs that exploits low-rank structure. We demonstrate that width, depth, and optimization criteria can be used as parameters related to model fidelity and show numerical justification of cost differences in training due to fidelity parameter choices. We test our multifidelity scheme on various canonical forward PDE models that have been presented in the emerging PINNs literature.

Keywords: Physics-Informed Neural Networks (PINNs), multifidelity, surrogate modeling, reduced-order modeling

1. Introduction

Engineering is replete with situations in which both low-fidelity (even "back of the envelope") models and high-fidelity models are available to aid in decision-making. It is often the case that the discrepancy between low-fidelity and high-fidelity is associated with a corresponding difference in computational cost: the low-fidelity simulation is far cheaper to compute than a high-fidelity simulation. These situations have motivated extensive research into multifidelity methods: those methods that attempt to judiciously combine low-fidelity and high-fidelity simulation results in an accuracy-increasing, cost-saving way. A comprehensive review of recent advances and works in the area of multifidelity methods can be found in the manuscript by Peherstorfer et al. [1].

Multifidelity construction of surrogate models for a diverse range of systems have been successfully implemented and presented in the literature. In many cases a well-established continuous relationship with respect to a discretization parameter exists, describing the convergence of the low-fidelity model to high-fidelity one. Hence, the notion of "fidelity" often represents a discrepancy of the low-fidelity model relative to the high-fidelity one, and accounts for amount of discretization coarsening, geometrical simplification, and underlying physical model complexity. Time-step size with theoretical guarantees used for canonical ODEs [2], time-step size in molecular dynamics simulation [3], quadratic frequency modulation in frequency-modulated trigonometric functions [4], finite element mesh size in acoustic horn problems [5], finite element mesh size in topological optimization problems [6, 7], finite volume discretization for heat driven cavity flow [8], aerodynamic model simplification in the parametric study of NACA airfoils [9] and amount of coarsening of the

Email addresses: mpenwarden@sci.utah.edu. (Michael Penwarden), zhe@cs.utah.edu (Shandian Zhe), akil@sci.utah.edu (Akil Narayan), kirby@cs.utah.edu (Robert M. Kirby)

Published in the Journal of Computational Physics

Eulerian and Lagrangian resolutions for the study of irradiated particle-laden turbulence [10] are among the examples of fidelity parameters used in the literature for the purpose of multifidelity construction of surrogate models. In all these cases, both high- and low-fidelity models attempt to model the same problem, and a direct relationship between the two can be perceived; the difference between the models in these cases is often a numerical discretization parameter that is chosen differently so that the low-fidelity model is less expensive, but also less accurate. Thus, in most of these cases the convergence of the low-fidelity model to the high-fidelity model can be proven analytically under refinement of discretization parameters and often there is a hierarchical connection between low- and high-fidelity models. For the cases of geometrical and physical simplifications, e.g., the composite beam example in [8], the low-fidelity model is indeed a simplified surrogate of the high-fidelity model, the latter of which includes more assumptions about the underlying physics of the system. Furthermore, one can even apply such methods when the fidelity is represented by the difference in quantized model hierarchies within discrete systems [11].

It is against this backdrop that we consider the extension of the low-rank parametric multifidelity approach of [4, 5] to Physics-informed Neural Networks (PINNs) [12, 13, 14]. PINNs represent a new "meshfree" discretization methodology built upon deep neural networks (DNNs), and capitalize on machine learning technologies such as automatic backward differentiation and stochastic optimization [15]. The marriage of computational modeling and machine learning is predicted to transform the way we do science, engineering and clinical practice [16].

In this paper, we adapt a multifidelity approach for parametric problems to PINNs, using the width and depth of the network architecture (for fixed activation functions) as well as optimization criteria as the means to determine fidelity levels. We provide theoretical discussions and experiments to motivate our width, depth and optimization criteria choices and their connection with fidelity. We also discuss some possible pitfalls of this connection. In regards to the connection between fidelity and computational cost: As the width and depth of a DNN is increased, the training time may increase significantly [15]. Hence we posit that PINNs is an admissible, if not ideal, candidate for multifidelity approaches.

The paper is organized as follows: In Section 2, we provide an overview of our low-rank multifidelity approach. In Section 3, we first review the original PINNs collocation approach and provide a brief summary of current and ongoing PINNs efforts within the field. Although we focus on the application of our multifidelity approach to the collocation version of PINNs, nothing precludes the extension of our work to other PINNs variants upon appropriate evaluation and minor modifications. In Section 4, we present our multifidelity PINNs approach applied to forward problems that have been presented in the emerging PINNs literature. Furthermore, we discuss the limitations and assumptions within our approach with present open theoretical and methodological challenges to the PINNs and machine learning communities. We conclude in Section 5 with a summary of our work and a discussion of current challenges and potential future avenues of inquiry and expansion of the concepts presented in this work.

2. Overview of our Low-rank Multifidelity Approach

Consider a low-fidelity and a high-fidelity model denoted, respectively, as,

$$g_L : D \to \mathbb{R}^m, \qquad\qquad g_H : D \to \mathbb{R}^M, \tag{1}$$

where $D \subset \mathbb{R}^d$ is a d-dimensional parameter space. In the context of parameterized partial differential equations (PDEs), we view g_L and g_H as solvers or emulators, mapping a common parameter space D to separate output spaces of different dimensions (e.g., as in coarse/fine or multiscale solvers). We make no assumptions, at this stage, on how m and M are related, but typically $m \ll M$. For a given parameter $p \in D$, we also make no explicit assumption about the physical meaning or interpretation of the model responses $g_L(p)$ versus $g_H(p)$; in particular we do not make formal assumptions ensuring $g_L(p) \approx g_H(p)$. In Section 3, we will provide some insights on how this setup can be realized in a PINNs framework.

We are interested in developing a multifidelity framework for PINNs, where g_L and g_H aim to model the same high-level system, but with different levels of accuracy or fidelity. In particular we assume the ordered hierarchy that g_H is a more expensive, but also more trusted predictor compared to g_L. For PINNs, we will use the width and depth of the network as well as optimization criteria, which are surrogates for expressivity and training optimality respectively, to define fidelity. In order to tackle this situation, our proposal is to use the methodology from [4], the use of which has subsequently been refined [5, 2, 17]. This procedure is relatively simple – containing the following high-level steps:

Step 1: Discretize parameter space D with $K \gg 1$ samples $D_L \coloneqq \{p_1, \ldots, p_K\}$.
Step 2: Evaluate g_L on the K points D_L, and use this to identify a subset of $k \ll K$ points $D_H = \{p_{i_1}, \ldots, p_{i_k}\} \subset D_L$ – the so-called "important" points in [4].
Step 3: Evaluate g_H at the k points in D_H.
Step 4: Construct a multifidelity emulator using stored low and high fidelity information at the "important" points.

Once these steps are completed, one has an emulator for the high-fidelity model constructed with the cost of dense sampling of the low-fidelity model and very few (k) high-fidelity samples. The evaluation of the emulator at any given parameter p costs a single low-fidelity model evaluation, but is an emulator for the high-fidelity model g_H. This is a low-rank procedure because the selection of points in Step 2, and the constructions in Step 4, exploits low-rank structure in certain matrices. We provide the details now.

The choice of D_L is problem-dependent; for example, K values uniformly sampled at random from the parameter space D is often used. Once the low-fidelity model g_L is evaluated at every point in D_L, a $K \times K$ Gram matrix G_L is constructed with the following entries,

$$(G_L)_{i,j} \quad = \quad \langle g_L(p_i), g_L(p_j) \rangle \quad i, j = 1, \ldots, K, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ is often chosen as the standard Euclidean inner product on the low-fidelity output space $\mathbb{R}^m$, corresponding to a *linear* kernel on two input features. Alternative inner product definitions, or choices of kernels, can be used depending on the properties of the Gramian that are desired; Razi et al. studied the impact of different kernels in comparison to the standard linear kernel above [18].

Note that the above is identical to formation of G_L via

$$G_L = V_L^T V_L, \qquad\qquad V_L \coloneqq \begin{pmatrix} g_L(p_1) & g_L(p_2) & \cdots & g_L(p_K) \end{pmatrix} \in \mathbb{R}^{m \times K}. \tag{3}$$

The procedure by which the k "important" points D_H are identified from D_L utilizes the matrix G_L (or, equivalently, V_L). Selecting "important" rows/columns from a matrix is a matrix subset selection problem that is related to low-rank approximation.

There are a number of procedures that provides selection tools for most important k indices from a Gramian matrix. Among the most commonly used selection approaches in the literature include linear algebraic strategies, such as pivots chosen by a column-pivoted QR decomposition of V_L [4], or equivalently the pivoted Cholesky decomposition of the Gram matrix G_L [5], the LU factorization [19]; statistical strategies such as leverage score sampling methods [20, 21]; and sparsity-promoting group matching methods [22, 21]. In this paper, we choose to use pivots identified from a pivoted Cholesky decomposition for this purpose: the procedure is easy to understand, readily implemented, available on many computational platforms, computationally efficient, and in our experience performs competitively with alternative methods [21]. The Cholesky approach forms the following pivoted Cholesky decomposition

$$P^T G_L P = R^T R, \; R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}$$

where R_{11} is a square matrix, and $P \in \mathbb{R}^{K \times K}$ is a permutation matrix whose entries identify column pivot indices in the decomposition process. We label these indices as $i_1, \ldots, i_K$, which are a permutation of the set $(1, \ldots, K)$.

It is well-known that the pivots chosen in this way are equivalent to those produced from a column-pivoted QR decomposition of V_L [23]. We have framed this discussion in the context of storing the full matrix G_L and computing the superfluous R_{11} and R_{12} factors, but one can construct algorithms that need not store the potentially large matrix G_L, and compute only the pivots in P; see, e.g., [4].

The multifidelity procedure chooses D_H from the pivots identified above:

$$D_H = \{p_{i_1}, \ldots, p_{i_k}\}.$$

The number k, representing the number of high-fidelity model evaluations that must be run, is chosen based on the available computational budget for the high-fidelity model. Because of the expense of the high-fidelity model, we frequently have that k is $\mathcal{O}(10)$ in practice. Next, we compute the high-fidelity model $g_H(p_{i_l})$ at these points; in situations of interest, this k-fold query of the high-fidelity model is typically the most expensive step of the procedure. Finally, the multifidelity approximation $\widetilde{g}_H$ can be constructed from these simulations:

$$g_H(p) \approx \widetilde{g}_H(p) \coloneqq \sum_{l=1}^{k} g_H(p_{i_l})\, c_l(p), \tag{4}$$

where $\{c_l(p)\}_{l=1}^k$ are coefficients computed via a least-squares projection from the low-fidelity model:

$$\begin{pmatrix} (G_L)_{i_1,i_1} & \cdots & (G_L)_{i_1,i_k} \\ \vdots & \ddots & \vdots \\ (G_L)_{i_k,i_1} & \cdots & (G_L)_{i_k,i_k} \end{pmatrix} \begin{pmatrix} c_1(p) \\ \vdots \\ c_k(p) \end{pmatrix} = \begin{pmatrix} \langle g_L(p), g_L(p_1)\rangle \\ \vdots \\ \langle g_L(p), g_L(p_k)\rangle \end{pmatrix}. \tag{5}$$

The coefficients $c_j(p)$ correspond to weights in a least-squares approximation of $g_L(p)$ using the basis $\{g_L(p_{i_j})\}_{j=1}^k$. The multifidelity technique therefore computes least-squares coefficients from the low-fidelity model, and uses these coefficients in the high-fidelity prediction. The coefficients $\{c_l(p)\}_{l=1}^k$ can also be computed directly from low-fidelity snapshots as,

$$c(p) = V_K^{\dagger} g_L(p) \in \mathbb{R}^K, \qquad V_K \coloneqq (g_L(p_{i_1}), \ldots, g_L(p_{i_K})) \in \mathbb{R}^{m \times K},$$

where $A^{\dagger}$ is the Moore-Penrose pseudoinverse of A [24]. An alternative to this approach would use least-squares to project the full low-fidelity ensemble onto the k important points, involving an $(m \times k)$ version of (5).

Because (5) is a square system, the approximation $\widetilde{g}_H$ is interpolatory at the important points: $\widetilde{g}_H(p) = g_H(p)$ for every $p \in D_H$. The evaluation of $\widetilde{g}_H$ can be accomplished with a single evaluation of the low-fidelity model g_L, which is required to form the right-hand-side of (5).

The overall accuracy of this approach, i.e., the efficacy of (4), depends on the discrepancy between G_L and G_H, but the actual model responses g_L and g_H need not have similar outputs [4]. Thus snapshot proximity, i.e., $g_L(p) \approx g_H(p)$, is *not* necessary for success of this procedure. Instead, we require a more subtle condition that the parameter variation of g_L is similar to that of g_H. See [4] for details and theoretical analysis. A practical procedure to estimate the error of this approach is provided in [8], and in Section 3.3.1 we provide futher discussion regarding the accuracy of (4).

Remark 1. In this paper, we provide bi-fidelity results as they give the starkest difference in levels. This is in part due to the well-known accuracy limit of PINNs. However, there is no reason more levels cannot be added. One would simply use the bi-fidelity method shown here in sequence to go through the intermediary fidelities. See [4] for a description of the multi-fidelity procedure with more than two levels.

3. Physics-Informed Neural Networks (PINNs)

In this section, we first present a review of Physics-Informed Neural Networks (PINNs), with an emphasis on the original collocation PINNs approach which we use in this work. We also provide a brief summary of current and ongoing PINNs efforts within the field – many if not all of which might benefit from the multifidelity approach presented herein. We then present the application of our proposed multifidelity approach to PINNs. We first provide a summary of the theoretical considerations upon which our work is built, and subsequently we provide a summary of the implementation considerations that are required.

3.1. Review of Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) were originally proposed by Karniadakis and co-workers [12, 13, 14] as a neural network based alternative to traditional PDE discretizations. In the original PINNs work, when presented with a PDE specified over a domain Ω with boundary conditions on $\partial\Omega$ and initial conditions at $t = 0$ (in the case of time-dependent PDEs), the solution is computed (i.e. the differential operator is satisfied) as in other mesh-free methods like RBF-FD [25, 26] at a collection of collocation points. First, we re-write our PDE system in residual form as $R(u) = f - \mathcal{N}(u)$ for an arbitrary differential operator $\mathcal{N}(u)$ which may be a function of both space and time. The PINNs formulation is expressed as follows: Let $\tilde{u}(x, t; \underline{\mathrm{w}})$ denote a neural network predictor with inputs (x, t) and parameters/weights $\underline{\mathrm{w}}$ that are the degrees of freedom of the network collected from its associated width and depth. In this paper, we assume $\underline{\mathrm{w}}$ is flattened and represented as a vector. Throughout the discussion, the activation function of the network is given and fixed. The network is trained based finding the weights $\underline{\mathrm{w}}$ that minimize the loss function:

$$MSE = MSE_u + MSE_R \tag{6}$$

where

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} \|\tilde{u}(x_u^i, t_u^i; \underline{\mathrm{w}}) - u^i\|^2 \tag{7a}$$

$$MSE_R = \frac{1}{N_R} \sum_{i=1}^{N_R} \|R(\tilde{u}(x_R^i, t_R^i))\|^2 \tag{7b}$$

where $\{x_u^i, t_u^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(x, t)$ and $\{x_R^i, t_R^i\}_{i=1}^{N_R}$ specify the collocation points for evaluation of the collocating residual term $R(\tilde{u})$. The loss MSE_u corresponds to the initial and boundary data while MSE_R enforces the structure imposed by the differential operator at a finite set of collocation points.

Beyond the initial collocation version of PINNs expressed above, Karniadakis and collaborators have extended these methods to conservative PINNs (cPINNs) [27], variational PINNs (vPINNS) [28], parareal PINNs (pPINNs) [29], stochastic PINNs (sPINNs) [30], fractional PINNs (fPINNs) [31], LesPINNs (LES PINNs) [32], non-local PINNs (nPINNs) [33] and eXtended PINNs (xPINNs) [34].

In this work, we will focus on application of the original collocation PINNs approach; however, the work presented herein can be applied to many if not all of these variants.

3.2. Expressivity of neural networks

PINNs are a special type of neural network (NN) formed from compositions of affine maps and componentwise activation functions where *depth* is the number of compositions and *width* is the number of intermediate outputs in each layer. A PINN regressor $\tilde{u}$ attempts to emulate the map $(x, t) \mapsto u(x, t)$. Solutions to nonlinear PDEs over large spatial and/or time scales result in very complex behavior of u as a function of the inputs (x, t), and the *expressivity* of a NN allows us to understand the NN's theoretical potential for faithfully predicting such complex behavior. When the activation function of the NN is piecewise linear, such as with the ReLU activation function, then the output $\tilde{u}$ is a piecewise linear function of (x, t), and the expressivity of an NN in this case is often measured as the number of (x, t) connected geometric regions over which $\tilde{u}$ is linear.

Consider a fully connected NN with L hidden layers each of width W. Many recent studies have established that it is possible for the number of linear regions in such an NN to grow proportionally to W^L [35, 36]. I.e., growth is exponential in the depth and algebraic in the width. This implies that the NN can theoretically express functions of exponentially growing complexity as the number of layers is increased, or of algebraically growing complexity by increasing width. Futher refinements of this theme have gained traction in the NN literature, including results that articulate expressive limitations with increasing depth [37, 38, 39, 40].

3.3. *A multifidelity method for Physics-Informed Neural Networks*

The theoretical expressivity of NN's provides one way to understand an accuracy-efficiency tradeoff with PINNs: NN architectures with larger width and/or depth have the potential to capture more complex PDE solutions, but this larger architecture is also more expensive to train through backpropagation and can require larger N_R and N_u in (7). Viewed in this light, a natural way to define fidelity for PINNs is through the network architecture. High-fidelity PINNs are those that have large width and/or depth, and low-fidelity PINNs have smaller width and depth.

The above interpretation of the accuracy-efficiency tradeoff as a notion of fidelity is based on theoretical insights about expressivity, but the practical realities surrounding expressivity are more complex: There is a large gap between the theoretical expressive potential of NN's and the realized expressivity of trained NN's in a practical setting. In particular, the exponentially expressive potential of increasing-depth networks is not frequently observed in practice, and with appropriate probabilistic models for weights of the network the average expressivity behaves linearly with respect to network depth [41].

Acknowledging that a practical training procedure that ensures monotonic accuracy with respect to NN width/depth and/or optimization criteria is not yet readily available, we posit that PINNs that are *low-fidelity* have simpler (less expressive) network architectures, and have looser optimization criteria. In contrast, *high-fidelity* PINNs have more complex architectures (higher expressivity) and stricter optimization criteria.

In this way, we can form a bi-level fidelity hierarchy with PINNs emulators. Let $\tilde{u}_L(x,t;\underline{w}_L)$ denote a trained low-fidelity PINN, and let $\tilde{u}_H(x,t;\underline{w}_H)$ denote a trained high-fidelity PINN. Here $\underline{w}_L$ and $\underline{w}_H$ denote the weight vectors associated to the high- and low-fidelity network architectures and optimization criteria and as such $\dim \underline{w}_L \ll \dim \underline{w}_H$. Since $\tilde{u}_L$ is less expressive than $\tilde{u}_H$, it should require less training time but also suffers from limited predictive power. The low-fidelity PINN $\tilde{u}_L$ is also trained with looser optimization criteria than $\tilde{u}_H$. Despite the practical caveat about tying architecture and optimization criteria to PINNs accuracy, we observe that increasing width and depth and strengthening optimization criteria does deliver a more accurate PINNs emulator, see Table 1 in Section 4.5. The particular choices of our network width, depth, and optimization criteria are provided in sections 3.3.3 and 4.

In the parametric multifidelity problem of Section 2, in order to construct a PINN for every parameter in the discretized parameter space $D_L = \{p_1, \ldots, p_K\}$, we must train K PINN's solutions. This is perhaps feasible for $\tilde{u}_L$, whose architecture is simpler and optimization criteria are looser, but not for the more expensive u_H. To address this difficulty, we employ a multifidelity procedure.

Let $\Xi \subset \Omega \times [0,T]$ denote the spatiotemporal domain of the PDE. (We present the methodology in the general spatiotemporal case; for a stationary PDE problem, we set $\Xi = \Omega$.) We let $\widetilde{\Xi}$ be a set of test points for the PINN, i.e., a size-P discretization of $\Omega \times [0,T]$ that are distinct from the PINN training set (i.e. collocation points) in this paper. Then the parameter models g_L and g_H are defined as the trained PINN at parameter values p evaluated on the grid $\widetilde{\Xi}$,

$$g_L(p) := (\tilde{u}_L(x_i,t_i;\underline{w}_L(p)))_{i=1}^P \in \mathbb{R}^P, \qquad g_H(p) := (\tilde{u}_H(x_i,t_i;\underline{w}_H(p)))_{i=1}^P \in \mathbb{R}^P, \tag{8}$$

where $\underline{w}_L(p)$ and $\underline{w}_H(p)$ are the weights computed from training the low- and high-fidelity PINN, respectively, and the points (x_i,t_i) are the spatiotemporal points that comprise $\widetilde{\Xi}$,

$$\widetilde{\Xi} = \{(x_i,t_i)\}_{i=1}^P . \tag{9}$$

Note that in this context, $g_L(p)$ and $g_H(p)$ are both vectors in P-dimensional space, so that in the notation of Section 2 we have $m = M = P$. While one could choose different test grids for u_L and u_H, we typically take $\widetilde{\Xi}$ to be a relatively dense and equispaced grid so that with an appropriate normalization the linear kernel/inner product produces an approximation to a continuous L^2 inner product,

$$\langle g_L(p), g_L(q)\rangle \approx \int_{\Omega\times[0,T]} \tilde{u}_L(x,t;\underline{w}_L(p))\tilde{u}_L(x,t;\underline{w}_L(q))dxdt, \tag{10}$$

and similarly for the high-fidelity model g_H.

Given this multifidelity surrogate on weights in high-fidelity space, we generate a multifidelity PINN (which we call mfPINN) with high-fidelity expressivity from the procedure in Section 2:

$$\widehat{u}_H(\widetilde{\Xi}; p) \coloneqq \widetilde{g}_H(p), \tag{11}$$

which is an approximation to $\tilde{u}_H(\widetilde{\Xi}; \underline{\mathrm{w}}_H(p))$ that is produced without the need to train $\underline{\mathrm{w}}_H(p)$. We recall that computing $\widetilde{g}_H(p)$ requires the single low-fidelity evaluation $g_L(p)$, i.e., a single training of $\tilde{u}_L$ at parameter p. If the cost of training $\tilde{u}_H$ is much larger than the cost of training $\tilde{u}_L$, then an accurate surrogate $\widehat{u}_H(p)$ can therefore be delivered with the training cost of only $\tilde{u}_L$.

Remark 2. The current state-of-the-art theory in DNNs provides a priori consistency statements about PINN solutions. However, a complete convergence theory remains as a topic of future work. Correspondingly, the results shown in this paper do not imply that every increased combination of depth and/or width of the network leads to improved accuracy of the DNN solution. The interplay of fidelity (expressivity) and optimization choices is a current area of research.

3.3.1. Theoretical Considerations

Theoretical analysis for the particular low-rank multifidelity surrogate of Section 2 is developed in [4, 2, 8]. The core requirements are that (i) the manifold of vectors $\{g_H(p) \mid p \in D\}$ is low-rank, and (ii) that one has an inner product proximity statement of the form,

$$\langle g_L(p), g_L(q)\rangle \approx \langle g_H(p), g_H(q)\rangle \,, \tag{12}$$

where $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product on vectors as in (2) and p and q denote positions within the sampled parameter space. In particular, it is *not* necessary that $g_L(p)$ be a good approximation to $g_H(p)$. Instead, one requires similar *parametric* dependence in the maps $p \mapsto g_L(p)$ and $p \mapsto g_H(p)$. We recall that with the dense grid $\widetilde{\Xi}$, then the inner product acting on evaluations of g_L approximates the continuous $L^2(\Omega \times [0, T])$ inner product, cf. (10). Under the assumption that $\tilde{u}_L$ and $\tilde{u}_H$ are approximations of appropriate accuracy to the true PDE solution u, then analysis can be carried out by exploiting (12) and the relationship to continuous L^2 norms (10) to conclude that the multifidelity procedure converges. For example, such an analysis exploits small Kolmogorov n width of the manifold induced by g_H [4], and can utilize accuracy estimates of $\tilde{u}_L$ and $\tilde{u}_H$ relative to u [2]. Many of the previously mentioned analysis techniques are theoretical in nature and can be difficult to apply in complex settings. Alternatively, the approach in [8] provides a computational strategy that produces model-independent algorithmic bounds on quantities similar to (12) that lead to estimates for error of $\widetilde{g}_H$.

3.3.2. Implementation Considerations

The multifidelity approach laid out in Section 2 contains four steps, summarized as: (1) sample the parameter space; (2) evaluate the low-fidelity model at the aforementioned samples and decide at which locations one should run the more costly high-fidelity model; (3) evaluate the high-fidelity model on the subset of "important" points; and lastly (4) use the set of low- and high-fidelity simulations to construct a multifidelity emulator. This multifidelity emulator, when queried at a new (not previously evaluated) location within the parameter space requires the evaluation of only the low-fidelity model and the new location combined with a computationally efficient manner of augmentation (which is significantly cheaper than evaluation of the high-fidelity model outright) to provide an updated fidelity response.

Based upon the theoretical considerations presented above, only Step (2) in the multifidelity procedure requires systematic modification. Implicit in Step (2) is the decision as to what fidelity means in the context of this approach and how is it tuned. In the case of PINNs, we have selected width and depth of the neural network architecture (with fixed activation functions) as the our adaptable fidelity hyperparameters.

3.3.3. Algorithmic Complexity

To aid in evaluating the algorithmic complexity and reproducibility of our approach, we provide Algorithm 1. Note that the first three steps: storing the p^C weights, storing trained low-fidelity solutions, and storing trained high-fidelity solutions, are offline steps. The online step is generating a new low-fidelity solution and computing the coefficients in Equation 5 for the multifidelity approximation. For this algorithm we define the notation for the centroid of domain D as p^C. Given the parametric PDE domain bounds $[a_i, b_i]$ in dimension i

$$p^C = \frac{a_i + b_i}{2}, \qquad i = 1, ..., d$$

The main considerations when evaluating the cost of our approach are the sizes K and k of the low and high fidelity sample sets respectively, and the computational costs of evaluating g_L and g_H associated with a PDE. Given that it is often the case that cost of g_H is much greater than the cost of g_L, minimizing k provides the greatest cost savings in our approach.

Algorithm 1 Multifidelity PINNs Method

```
Given appropriate set-up information (i.e. domain discretization $\tilde{\Xi}$, low-fidelity parameter space sampling $D_L$, PDE initial and boundary conditions, low & high-fidelity sample sizes $K$ & $k$, PINN hyper-parameters (size and optimization) for $\tilde{u}_L$ & $\tilde{u}_H$ etc.)
Assume $\frac{||u-\tilde{u}_L||_2}{||u||_2} > \frac{||u-\tilde{u}_H||_2}{||u||_2}$
Assume $\tilde{u}_L$ *training time* $<$ $\tilde{u}_H$ *training time*
Store trained weights $\underline{w}_L(p^C)$ from $\tilde{u}_L(\Xi; \underline{w}_L(p^C))$
for n = 1 to $K$ do
  Initialize $\tilde{u}_L$ weights $\underline{w}_L(p_n)$ with $\underline{w}_L(p^C)$
  Store trained $\tilde{u}_L(\tilde{\Xi}; \underline{w}_L(p_n))$ solutions
end for
Compute Gram matrix $G_L$ as defined in (2)
Set $D_H$ by selecting $k$ points in $D_L$ identified from the first $k$ pivots in the pivoted Cholesky decomposition of $G_L$ as described in Section 2
Store trained weights $\underline{w}_H(p^C)$ from $\tilde{u}_H(\tilde{\Xi}; \underline{w}_H(p^C))$
for n = 1 to $k$ do
  Initialize $\tilde{u}_H$ weights $\underline{w}_H(p_{i_n})$ with $\underline{w}_H(p^C)$
  Store trained $\tilde{u}_H(\tilde{\Xi}; \underline{w}_H(p_{i_n}))$ solutions
end for
To use multifidelity method: solve $\tilde{u}_L$ at any point $p$, then perform the multifidelity procedure in (4) & (5) using the stored $\tilde{u}_L$ & $\tilde{u}_H$ solutions to construct $\widehat{u}_H(\tilde{\Xi}; p)$ which emulates $\tilde{u}_H(\tilde{\Xi}; \underline{w}_H(p))$
```

Remark 3. The optimization process for DNNs is highly dependent on the initial state of the weight matrix that is used [15]. The non-convex nature of the optimization problem often leads to many local minima, all of which have very similar loss function evaluations. In some applications like metalearning of PINNs, smooth transitions in the weight matrices across parameterized runs are desired [42]. In this work, we have initialized our weight matrices by first computing a PINN solution at the midpoint of the parametric domain and then using that weight matrix as initialization for all subsequent parametric point evaluations. This is known to reduce not only cost but also reduce the likelihood that a PINN will not be trained well, which is important in the online step.

In regards to optimization, all runs (both low and high) are initially optimized with 500 epochs using the Adam version of stochastic gradient decent (SGD) with a learning rate of 10^{-3}. This is to help get in the vicinity of a loss function minimum before using the more precise L-BFGS optimizer, which has been shown to aid the consistency of optimizing DNNs as well as PINNs. Without doing employing this strategy, we encountered more “bad” runs where using *only* L-BFGS does not optimize well.

In the context of optimization criteria and in terms of L-BFGS, we specify that for low-fidelity runs the maximal number of iterations per optimization step is 5000, the termination tolerance on a first order optimality condition is 10^{-6}, and the termination tolerance on weight vector changes is 10^{-9}. For high-fidelity runs, these are $10,000$, 10^{-9}, and 10^{-12} respectively. We found that this helps increase the accuracy and time differences between low and high-fidelity which give more distinct results. (We again emphasize that PINNs, and DNNs in general, do not in practice always yield higher accuracy with a more expressive architecture even with a very large amount of data.) As mentioned earlier, as a pre-optimization step, we initialize the weights of the PINN with the final weights of a PINN run at the center of the PDE hyper-parameter range for low and high respectively.

4. Results and Discussion

In this section, we demonstrate the efficacy of our multifidelty approach on four forward PDE problems in one and two spatial dimensions: 1D Burgers' equation, a 1D nonlinear heat equation, the 2D nonlinear Allen-Cahn equation, and a 2D nonlinear Diffusion-Reaction equation. These examples are extensions of the test problems proposed in [12, 43]. For all the PINNs architectures used below, we employ fully-connected feed-forward neural networks with tanh activation functions. There are two types of snapshots spaces here: (i) the Burgers' and (nonlinear) Heat equations in which we have a spatial and temporal dimension, and the (ii) Allen-Cahn and Diffusion-Reaction equations in which we have two spatial dimensions. For the former, the test set of points $\widetilde{\Xi}$ is a uniform grid of size 256×100 in space and time respectively. For the latter we use a uniform grid of size 128×128. These test points are where we compare between the exact solution and the PINN solution. To train the PINN we use 100 uniformly sampled boundary/initial value points for the MSE_u portion of the loss, and $10,000$ collocation points using Latin hypercube sampling (LHS) for the MSE_R portion as described in Section 3.

4.1. 1D Burgers Equation

We consider the following 1D viscous Burgers equation:

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial}{\partial x}\left(u^2\right) = \nu \frac{\partial^2 u}{\partial x^2} \tag{13}$$

on $(x, t) \in \Xi = [-1, 1] \times [0, 1]$ with the viscosity $\nu \in [0.005, 0.05]$ and initial condition $u(x, 0) = -\sin(x\pi)$. Our parameter in this example is the viscosity, i.e., $p = \nu$. For evaluation of the error, we compute the exact solution derived using Cole's transformation computed with Hermite integration [44]. Our sampled parameter space D_L is constructed with $K = 50$ LHS points, and the low-fidelity PINN $\tilde{u}_L$ has 2 hidden layers each of width 5. Based upon our multifidelity procedure, we evaluate $k = 10$ high-fidelity PINNs samples, chosen by the pivoted Cholesky decomposition of the low-fidelity snapshot matrix. The high-fidelity PINN $\tilde{u}_H$ consists of 5 hidden layers of width 10. Using the collected low- and high-fidelity ensembles, we can perform the multifidelity procedure on any future low fidelity runs to evaluate the multifidelity emulator $\widehat{u}_H$. For this case with one PDE parameter in the Burgers' equation, we generate test points over 100 uniformly sampled points and compare to the exact solution to generate the plot in Figure 13.

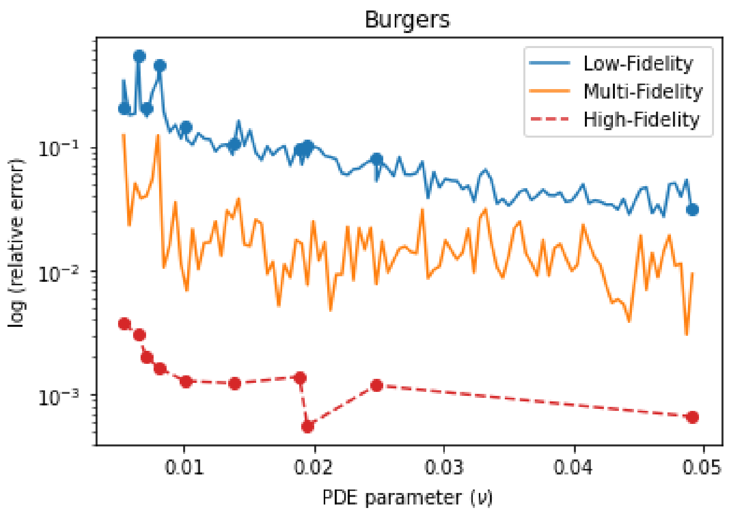

Figure 1: Line plot of the log (base 10) relative error between the exact solution and PINN solution at a given parameter, in this case viscosity as defined in Equation 13. Note that the multifidelity emulator error is sandwiched between the low and high fidelity runs, as would be expected. This behavior is not guaranteed if the high-fidelity PINNs are inaccurate compared to the low-fidelity ones, but this depends on the formulation of the problem attempted. If it is a well posed problem with well chosen parameter ranges, low and high-fidelity DNN sizes, etc. then this behavior is frequently observed.

The results in Figure 1 show that low fidelity accuracy can be enhanced with this multifidelity procedure, if one invests in training a modest number of higher-fidelity PINNs. We also observe that the relative error decreases with increasing viscosity, which aligns with expectations that small viscosity corresponds to regimes where shocks can form, which are generally more difficult to approximate. The red and blue dots represent the $k = 10$ important parameters locations selected by the multifidelity procedure. We observe that the pivoted Cholesky decomposition clusters points toward lower viscosity, which again agrees with expectations since in that near-shock regime the parametric variations are more complex. Lastly, the noisy fluctuations can be explained since DNNs have substantial variance in their solutions due to randomness imparted during training (e.g., with Adam), so the accuracy of these solutions does not vary smoothly with PDE parameter. We attempt to reduce these fluctuations using previously described methods, such as running Adam before L-BFGS and initializing the weights using a previous PINN solution at the parameters center.

4.2. 1D nonlinear Heat Equation

We consider the following 1D nonlinear PDE:

$$\frac{\partial u}{\partial t} - \lambda \frac{\partial^2 u}{\partial x^2} + k \tanh(u) = f, \, x \in \Omega \tag{14}$$

where $(x, t) \in \Xi = [-1, 1] \times [0, 1]$ and where $\lambda \in [1, \pi]$ and $k \in [1, \pi]$ are positive constants. Our parameter in this example will be the joint tuple $p = (\lambda, k)$. In order to compute errors, we employ the method of manufactured solutions, specifying an exact solution of $u(x, t; \lambda, k) = k \, sin(\pi x) \, exp(-\lambda k x^2) \, exp(-\lambda t^2)$ and derive the corresponding form of the forcing f. We choose D_L as $K = 50$ LHS samples in parameter space. The low-fidelity PINNs have 2 hidden layers each of width 5. Based upon our multifidelity procedure, we choose $k = 10$ important parameter locations and evaluate that many high-fidelity PINNs. The high-fidelity PINN has an architecture of 5 hidden layers of width 10. For this case with two-dimensional parameter space, we use an equidistant 20×20 grid as the set of test points.

Results are shown in Figure 2, where the shown surface plots correspond to a 100×100 grid that is generated via cubic spline interpolation from the 20×20 test grid. Note that the cubic spline procedure visually smooths out noisy fluctuations, cf. Figure 1. The multifidelity procedure clearly improves accuracy in the parameter region chosen, and we can see the general trend is that low k values and high λ values have higher errors for the low-fidelity PINN. We observe that the multifidelity procedure ameliorates this inaccuracy.

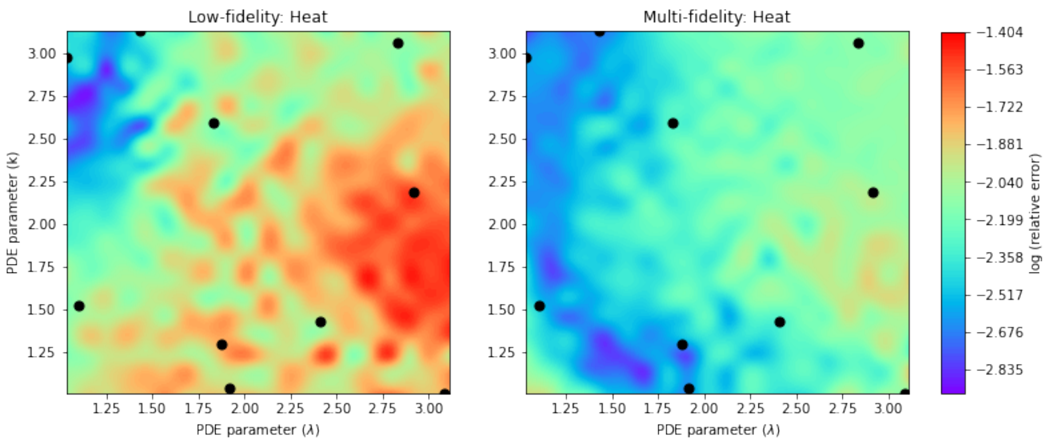

Figure 2: Surface plot of the log (base 10) relative errors between the exact solution and PINN solution at given parameters, in this case k and λ as defined in Equation 13. The black dots represent the parameter locations at which the multifidelity procedure was constructed with $k = 10$ high fidelity runs.

4.3. 2D nonlinear Allen-Cahn Equation

We consider the following 2D nonlinear Allen-Cahn equation, which is a widely used model for multi-phase flows:

$$\lambda\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+u\left(u^2-1\right)=f,\; x,y\in\Omega \tag{15}$$

where $\Xi = [-1,1]^2$, with $\lambda \in (0,\pi]$ the mobility, and u denotes the order parameter, prescribing different phases. The parameter in this problem is $p = \lambda$. We again use the method of manufactured solutions, specifying an exact solution of $u(x,y;\lambda) = \exp(-\lambda(x+0.7))\sin(\pi x)\sin(\pi y)$ and derive the corresponding form of the forcing f. Parameter space is discretized with $K = 50$ sampled uniformly at random. The low-fidelity PINN has 5 hidden layers of width 5, and the high-fidelity PINN has 8 hidden layers of width 10. We budget $k = 10$ high fidelity PINNs evaluations.

Remark 4. Both the low and high fidelity architectures are more complex and expressive than the ones used in the previous Burgers' and Heat equation examples. It is likely that the nature of the PDE being solved will strongly influence what sizes are appropriate for each fidelity. These architectural parameters were manually tuned to give reasonable results, and automatic procedures for such hyperparameter selection is the topic of ongoing investigations.

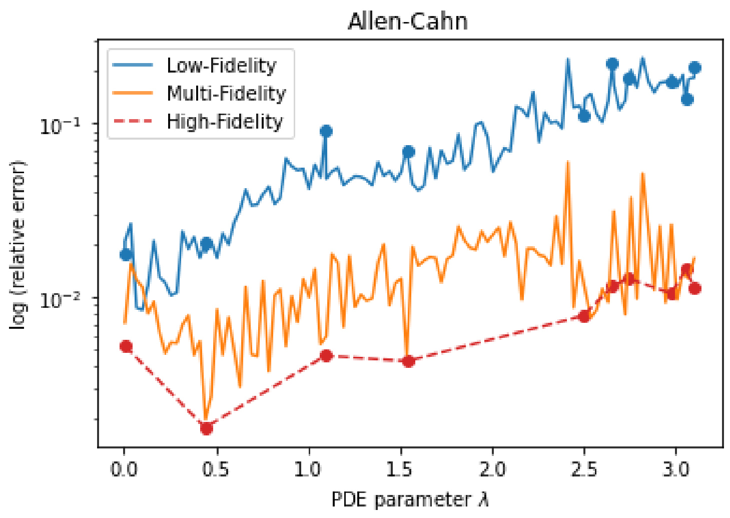

Figure 3: Log (base 10) relative error between the exact solution and PINN solution at a given parameter, in this case λ as defined in (15).

With this Allen-Cahn example, we observe the inverse relation between error and PDE parameter to that in Burgers. Here as λ increases, so does the solution complexity and therefore the error. We also observe the clustering of multifidelity "important points" in the region with the highest error.

Remark 5. Note that around $\lambda = 2.7$, we observe that the multifidelity approach has a lower error than the high fidelity points and considerable oscillations over the domain. The error of the multifidelity procedure is not strictly bounded by the low and high fidelity methods: There is a considerable variance when training neural networks as the process itself is randomized in the initialization and/or optimization. Each time a network is trained, the results may be slightly different even at the same parameter locations. This stochasticity implies that we cannot ensure that the multifidelity error is strictly bounded by the low- and high-fidelity errors. The oscillations/variance observed are also a direct result of this stochasticity.

4.4. 2D nonlinear Diffusion-Reaction Equation

We consider the following 2D nonlinear diffusion-reaction equation:

$$\lambda\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+k\left(u^2\right)=f,\ x,y\in\Omega \tag{16}$$

where $(x, y) \in \Xi = [-1, 1]^2$. Here $\lambda \in [1, \pi]$ represents the diffusion coefficient and $k \in [1, \pi]$ represents the reaction rate and f denotes the source term. Our parameter p is the tuple (λ, k). We specify an exact solution as $u(x, y; \lambda, k) = k\,\sin(\pi x)\,\sin(\pi y)\,\exp(-\lambda\sqrt{(k\,x^2 + y^2)})$ and derive the corresponding form of the forcing f. The low-fidelity PINN has 5 hidden layers of width 10, and the high-fidelity PINN has 8 hidden layers of width 20. Parameter space is discretized with $K = 50$ points drawn uniformly at random. Results of this experiment are shown in Figure 4. Similar to the Heat equation example, we observe that the multifidelity procedure improves accuracy compared to the low-fidelity PINN.

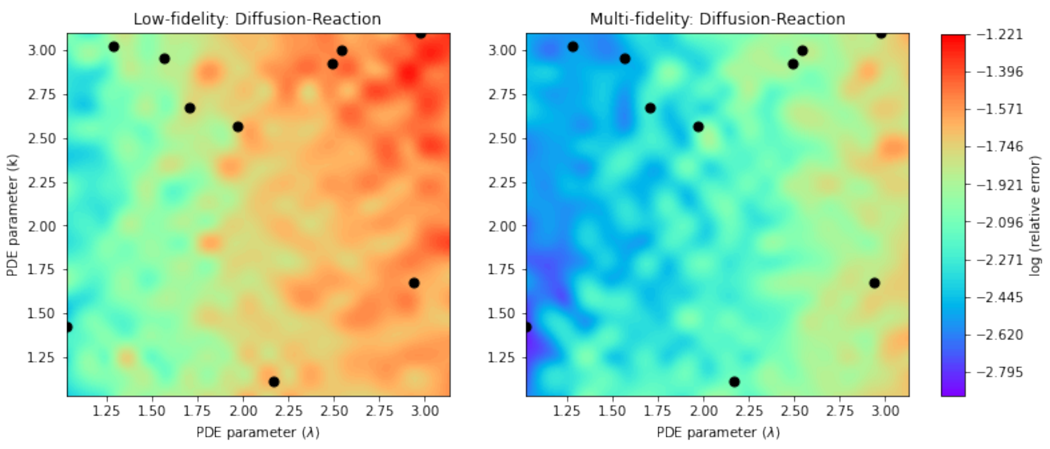

Figure 4: Surface plot of the log (base 10) relative error between the exact solution and PINN solution. The parameters are k and λ as defined in (16). The black dots show the $k = 10$ important points in parameter space selected by the pivoted Cholesky procedure.

4.5. Performance Metrics

We summarize numerical accuracy and cost measurements of the low-fidelity and high-fidelity PINNs, and of the multifidelity emulator in Table 1. These experiments were run on an Intel Core i7-5930K processor with the Windows 10 OS. PyTorch version 1.6.0 and Python version 3.8.5 were used to run the PINNs. Average values and standard deviations of cost and error computed over parameter space are reported.

Table 1: Metrics of performance taken for each of the four problems where $\pm$ indicates 1 standard deviation. For relative error the values presented are the mean of the PDE parameter 1D or 2D grids sampled as shown in the previous subsections. The computational time (in seconds) corresponds to training time in the L-BFGS portion of the optimization for the indicated PINNs.

PDE's	1D Burgers	1D Heat	2D Allen-Cahn	2D Diffusion-Reaction
Low-Fidelity Error (10^{-2})	7.86 ± 5.77	1.24 ± 0.69	7.71 ± 5.61	1.82 ± 0.91
Multi-Fidelity Error (10^{-2})	1.82 ± 1.76	0.55 ± 0.25	1.39 ± 0.92	0.81 ± 0.51
High-Fidelity Error (10^{-2})	0.17 ± 0.09	0.46 ± 0.25	0.85 ± 0.41	0.75 ± 0.51
Low-Fidelity Time (s)	41 ± 19	63 ± 12	167 ± 32	212 ± 13
High-Fidelity Time (s)	137 ± 72	181 ± 64	403 ± 137	890 ± 207

The low and high-fidelity PINNs correspond to less and more accurate solvers, respectively, and also correspond to less and more costly solvers, respectively. Since the multifidelity emulator cost equals that of the low-fidelity PINN, this demonstrates significant potential for savings in parametric multi-query contexts. The high-fidelity solver is around twice as expensive for the Allen-Cahn example, and around 4 times as costly for the Diffusion-Reaction example. In every test problem, the mean error of the multifidelity offers substantial improvement over its low-fidelity counterpart.

4.6. Convergence Plots

We provide Figure 5 to show the convergence of the example problems as more points are added in the multifidelity method, giving more insight into the accuracy-cost tradeoffs of our method. Our results in previous sections utilized a procedure corresponding to the blue line in Figure 5. In this figure, the labels "center" and "random" are in reference to the NN weight initialization; the first word refers to the initialization used for constructing the multifidelity method, and the second for generating the test points. E.g., "Center — Random" denotes that the multifidelity initialization is chosen according to the *center* run and that the error on test points is computed by constructing a PINN whose initialization is *random*. As briefly discussed in Remark 3 in Section 3.3.3, we initialize with the trained weights at p^C. For additional intuition about this, we refer the reader to [42]. Using this strategy, we not only observe a decrease in cost, but as seen in the convergence plots for all problems, the multifdelity method converges as well. By initializing in this way, we empirically obtain smooth NN weights in the parametric domain. It is then logical that in our multifidelity method, we observe better behavior due to less variance in the weights and, therefore, in the solution. However, we provide an experiment corresponding to the green line in Figure 5, where all initialization is random, to demonstrate that even with standard initialization approaches our method works well. We also show the orange line in Figure 5, showing a center initialization in training but a random initialization during the test phase, to demonstrate efficacy even if training uses a center initialization, but a different (random) initialization is used in the test phase. Finally, our method is shown to converge after only a few points even though we provide solutions with ten points so one could cut down on offline cost even more in practice.

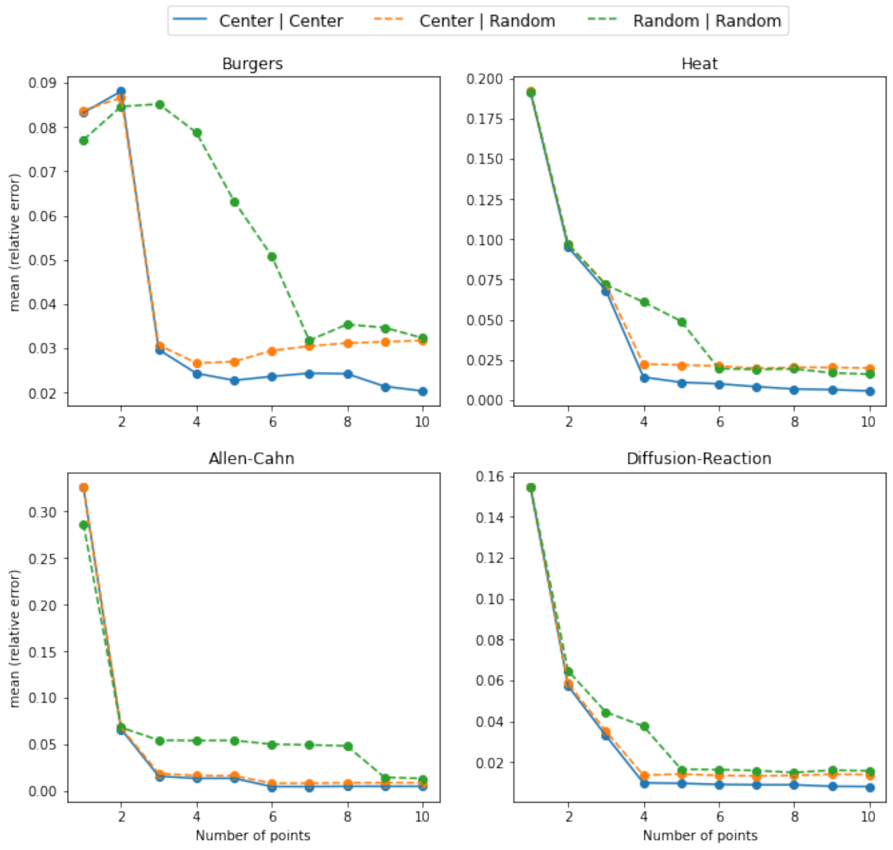

Figure 5: Convergence plots of the four PDE problems: Burgers, Heat, Allen-Cahn, Diffusion-Reaction. Three cases were run: one in which we initialize with the center of the parameter space for both training and testing (blue), one in which we initialize for training but not testing (orange), and one in which we use standard randomized weights for both steps (green). The multifidelity method converges regardless of this choice, but it is ideal to initialize for both steps, as seen by blue line

5. Summary and Conclusions

In this paper, we have extended the low-rank multifidelity approach of [4, 5] to Physics-informed Neural Networks (PINNs) [12, 13, 14]. Having summarized the multifidelity approach and the collocation version of PINNs, we construct a low-fidelity version of a PINN as one with a simpler (less expressive) architecture and less stringent optimization termination criteria. We empirically demonstrate that low-fidelity PINNs constructed in this way can be both less costly and less accurate than high-fidelity versions. This motivates our proposed multifidelity approach, whose computational results on several PDEs demonstrate that the multifidelity emulator can provide an accuracy-increasing PINNs surrogate over a PDE-based parameter space at significant savings in computational cost. We also observe that initializing the weights with a *center* run as in [42] helps to speed up convergence as a function of multifidelity points used.

Future investigations will explore more quantitative connections between PINNs architecture and accuracy/cost tradeoffs. In addition, the multifidelity approach considered here is agnostic to the type of PDE solver employed. We have used PINNs as both low and high-fidelity models. However, for example, one could utilize PINNs for a low-fidelity model and more traditional numerical solvers (such as finite element methods) for the high-fidelity model. Such combinations could generate a multifidelity surrogate that combines the advantages of different types of solvers.

Acknowledgements: The authors would like to acknowledge helpful discussions with Professor George Karniadakis and his group (Brown University). This work was funded under AFOSR MURI FA9550-20-1-0358. A. Narayan was partialy supported by AFOSR FA9550-20-1-0338.

Bibliography

References

[1] B. Peherstorfer, K. Willcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, SIAM Review 60 (3) (2018) 550–591.

[2] V. Keshavarzzadeh, R. M. Kirby, A. Narayan, Convergence acceleration for time-dependent parametric multifidelity models, SIAM Journal on Numerical Analysis 57 (2019) 1344–1368.

[3] M. Razi, A. Narayan, K. R. M., D. Bedrov, Fast predictive models based on multi-fidelity sampling of properties in molecular dynamics simulations, Computational Material Science 152 (C) (2018) 125–133.

[4] A. Narayan, C. Gittelson, D. Xiu, A stochastic collocation algorithm with multifidelity models, SIAM Journal on Scientific Computing 36 (2) (2014) A495–A521.

[5] X. Zhu, A. Narayan, D. Xiu, Computational aspects of stochastic collocation with multifidelity models, SIAM/ASA Journal on Uncertainty Quantification 2 (1) (2014) 444–463.

[6] V. Keshavarzzadeh, R. M. Kirby, A. Narayan, Parametric topology optimization with multi-resolution finite element models, International Journal on Numerical Methods in Engineering 119 (2019) 567–589.

[7] V. Keshavarzzadeh, M. Alirezaei, T. Tasdizen, R. M. Kirby, Image-based multiresolution topology optimization using deep disjunctive normal shape model, Computer-Aided Design 130 (2021) 102947.

[8] J. Hampton, H. R. Fairbanks, A. Narayan, A. Doostan, Practical error bounds for a non-intrusive bi-fidelity approach to parametric/stochastic model reduction, Journal of Computational Physics 368 (2018) 315–332.

[9] R. Skinner, A. Doostan, E. Peters, J. Evans, K. E. Jansen, An evaluation of bi-fidelity modeling efficiency on a general family of NACA airfoils, in: 35th AIAA Applied Aerodynamics Conference, 2017, p. 3260.

[10] L. Jofre, G. Geraci, H. Fairbanks, A. Doostan, G. Iaccarino, Multi-fidelity uncertainty quantification of irradiated particle-laden turbulence, arXiv preprint arXiv:1801.06062 (2018).

[11] M. Razi, R. M. Kirby, A. Narayan, Fast predictive multi-fidelity prediction with models of quantized fidelity levels, Journal of Computational Physics 376 (2019) 992–1008.

[12] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[13] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561 (2017).

[14] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations, arXiv preprint arXiv:1711.10566 (2017).

[15] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, Cambridge, MA, USA, 2016.

[16] M. Alber, A. Buganza Tepole, W. R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W. W. Lytton, P. Perdikaris, L. Petzold, E. Kuhl, Integrating machine learning and multiscale modeling: perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences, Nature: npg digital medicine 2 (115) 115.

[17] J. Hampton, H. R. Fairbanks, A. Narayan, A. Doostan, Practical error bounds for a non-intrusive bi-fidelity approach to parametric/stochastic model reduction, Journal of Computational Physics 368 (2018) 315–332, arXiv: 1709.03661. doi:10.1016/j.jcp.2018.04.015.
URL http://www.sciencedirect.com/science/article/pii/S0021999118302298

[18] M. Razi, R. M. Kirby, A. Narayan, Kernel optimization for low-rank multi-fidelity algorithms, International Journal for Uncertainty Quantification 11 (2021) 31–54.

[19] D. Anderson, M. Gu, An efficient, sparsity-preserving, online algorithm for low-rank approximation, in: International Conference on Machine Learning, 2017, pp. 156–165.

[20] D. J. Perry, R. T. Whitaker, Augmented leverage score sampling with bounds, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2016, pp. 543–558.

[21] D. J. Perry, R. M. Kirby, A. Narayan, R. Whitaker, Allocation strategies for high fidelity models in the multifidelity regime, SIAM Journal on Uncertainty Quantification 7 (1) (2019) 203–231.

[22] A. Lozano, G. Swirszcz, N. Abe, Group orthogonal matching pursuit for logistic regression, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 452–460.

[23] G. H. Golub, C. F. V. Loan, Matrix Computations, 3rd Edition, The Johns Hopkins University Press, 1996.

[24] D. Harville, The moore-penrose inverse, in: Matrix Algebra From a Statistician's Perspective, Springer, New York, NY., 1997, pp. 497–.

[25] V. Shankar, G. B. Wright, A. L. Fogelson, R. M. Kirby, A radial basis function (rbf)-finite difference method for the simulation of reaction-diffusion equations on stationary platelets within the augmented forcing method, International Journal for Numerical Methods in Fluids 75 (2014) 1–22.

[26] V. Shankar, G. B. Wright, R. M. Kirby, A. L. Fogelson, A radial basis function (rbf)-finite difference (fd) method for diffusion and reaction-diffusion equations on surfaces, Journal of Scientific Computing 63 (2014) 745–768.

[27] A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Computer Methods in Applied Mechanics and Engineering 365 (2020) 113028. doi:https://doi.org/10.1016/j.cma.2020.113028.
URL https://www.sciencedirect.com/science/article/pii/S0045782520302127

[28] E. Kharazmi, Z. Zhang, G. E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations (2019). arXiv:1912.00873.

[29] X. Meng, Z. Li, D. Zhang, G. E. Karniadakis, Ppinn: Parareal physics-informed neural network for time-dependent pdes, Computer Methods in Applied Mechanics and Engineering 370 (2020) 113250. doi:https://doi.org/10.1016/j.cma.2020.113250.
URL https://www.sciencedirect.com/science/article/pii/S0045782520304357

[30] D. Zhang, L. Lu, L. Guo, G. E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, Journal of Computational Physics 397 (2019) 108850. doi:https://doi.org/10.1016/j.jcp.2019.07.048.
URL https://www.sciencedirect.com/science/article/pii/S0021999119305340

[31] G. Pang, L. Lu, G. E. Karniadakis, fpinns: Fractional physics-informed neural networks (2018). arXiv:1811.08967.

[32] X. I. A. Yang, S. Zafar, J.-X. Wang, H. Xiao, Predictive large-eddy-simulation wall modeling via physics-informed neural networks, Phys. Rev. Fluids 4 (2019) 034602. doi:10.1103/PhysRevFluids.4.034602.
URL https://link.aps.org/doi/10.1103/PhysRevFluids.4.034602

[33] G. Pang, M. D'Elia, M. Parks, G. E. Karniadakis, npinns: nonlocal physics-informed neural networks for a parametrized nonlocal universal laplacian operator. algorithms and applications (2020). arXiv:2004.04276.

[34] A. Jagtap, G. Karniadakis, Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Communications in Computational Physics 28 (2020) 2002–2041. doi:10.4208/cicp.OA-2020-0164.

[35] R. Pascanu, G. Montufar, Y. Bengio, On the number of response regions of deep feed forward networks with piece-wise linear activations, arXiv:1312.6098 [cs]ArXiv: 1312.6098 (Feb. 2014).
URL http://arxiv.org/abs/1312.6098

[36] G. Montúfar, R. Pascanu, K. Cho, Y. Bengio, On the number of linear regions of deep neural networks, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, MIT Press, Cambridge, MA, USA, 2014, pp. 2924–2932, arXiv: 1312.6098.

[37] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, J. Sohl-Dickstein, On the Expressive Power of Deep Neural Networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 2847–2854, iSSN: 2640-3498.
URL http://proceedings.mlr.press/v70/raghu17a.html

[38] M. Telgarsky, Benefits of depth in neural networks, in: Conference on Learning Theory, PMLR, 2016, pp. 1517–1539, iSSN: 1938-7228.
URL http://proceedings.mlr.press/v49/telgarsky16.html

[39] M. Telgarsky, Representation Benefits of Deep Feedforward Networks, arXiv:1509.08101 [cs]ArXiv: 1509.08101 (Sep. 2015).
URL http://arxiv.org/abs/1509.08101

[40] T. Serra, C. Tjandraatmadja, S. Ramalingam, Bounding and Counting Linear Regions of Deep Neural Networks, arXiv:1711.02114 [cs, math, stat]ArXiv: 1711.02114 (Sep. 2018).
URL http://arxiv.org/abs/1711.02114

[41] B. Hanin, D. Rolnick, Complexity of Linear Regions in Deep Networks, arXiv:1901.09021 [cs, math, stat]ArXiv: 1901.09021 (Jun. 2019).
URL http://arxiv.org/abs/1901.09021

[42] M. Penwarden, S. Zhe, A. Narayan, R. M. Kirby, Physics-informed neural networks for parameterized pdes: A metalearning approach, Journal of Computational Physics Under Review (2021).
URL https://arxiv.org/abs/2110.13361

[43] L. Yang, X. Meng, G. E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse problems with noisy data, Journal of Computational Physics 425 (2021) 109913.

[44] C. Basdevant, M. Deville, P. Haldenwang, J. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, A. Patera, Spectral and finite difference solutions on the Burgers equation, Computers & Fluids 14 (1986) 23–41.