Version of Record: https://www.sciencedirect.com/science/article/pii/S0925231220301909
Manuscript_8c73bfbc80ffd0eab4cfe9252995a7a0

Solving Differential Equations using Deep Neural Networks

Craig Michoski[a,*], Miloš Milosavljević[b], Todd Oliver[a], David Hatch[c]

[a] *Oden Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin,TX 78712*
[b] *Department of Astronomy, The University of Texas at Austin, Austin, TX 78712*
[c] *Institute for Fusion Studies, University of Texas at Austin, Austin,TX 78712*

Abstract

Recent work on solving partial differential equations (PDEs) with deep neural networks (DNNs) is presented. The paper reviews and extends some of these methods while carefully analyzing a fundamental feature in numerical PDEs and nonlinear analysis: irregular solutions. First, the Sod shock tube solution to the compressible Euler equations is discussed and analyzed. This analysis includes a comparison of a DNN-based approach with conventional finite element and finite volume methods, and demonstrates that the DNN is competitive in terms of degrees of freedom required for a given accuracy. Further, the DNN-based approach is extended to consider performance improvements and simultaneous parameter space exploration. Next, a shock solution to compressible magnetohydrodynamics (MHD) is solved for, and used in a scenario where experimental data is utilized to enhance a PDE system that is *a priori* insufficient to validate against the observed/experimental data. This is accomplished by enriching the model PDE system with source terms that are then inferred via supervised training with synthetic experimental data. The resulting DNN framework for PDEs enables straightforward system prototyping and natural integration of large data sets (be they synthetic or experimental), all while simultaneously enabling single-pass exploration of an entire parameter space.

Keywords: Deep neural networks, differential equations, partial differential equations, nonlinear, shocks, data analytics, optimization

Contents

*Corresponding author
Email address: michoski@oden.utexas.edu (Craig Michoski)

1. Introduction

Solving differential equations using optimization/minimization strategies has been popular for some time in the form of least squares finite element methods (LSFEM) [29, 7, 57], and element-free Galerkin methods [4, 36], where even for mesh-free formulations, theoretical convergence criteria have been examined [3]. Approaches such as these were originally applied to neural networks in [32], though using neural networks in this context has recently experienced a resurgence of interest as seen in [5, 25, 54, 51, 45]. These recent works have shown that remarkably simple implementations of deep neural networks (DNNs) can be used to solve relatively broad classes of differential equations. From a method development point of view, demonstrating this capability is in and of itself of research interest. However, it remains unclear what the practical utility and scope of these types of methods may ultimately be within science and engineering applications, as well as for the numerical analysis of differential systems.

To explore how efficient, reliable, robust, and generalizable DNN-based solutions are will undoubtedly require careful and prolonged study. Here, we offer a practical starting point by asking how well, and in what ways, can DNNs manage one of the cleanest, simplest, and most ubiquitous irregularities observed in differential systems, namely shock fronts. Shock fronts provide a quintessential test bed for a numerical method in that they contain an isolated analytically non-differentiable feature that propagates through the differential system. The feature reduces the local regularity of the solution both in space and time and leads to numerical representations that must accommodate local first-order analytic discontinuities while still maintaining some concept of numerical stability and robustness. Indeed the way a numerical method responds to a local shock front tends to reveal the fundamental signature response that the numerical method has to local irregularities in the solution, which might be thought of as providing an indicator by which to gauge the method's applicability in such cases.

A tremendous amount of work spanning pure mathematical analysis [33, 55], numerical analysis [1, 46], physics [10, 61], etc., has been done on differential systems that demonstrate shock-like behavior. Shock-like numerical behavior is deeply rooted in discrete function representation and approximation theory, and further in functional and discrete functional analysis, specifically in what it means to locally and globally approximate a member of a function space [26, 19, 18, 11].

Along these lines, the main portion of this paper is dedicated to examining the method of DNNs for solving PDEs, and exploring the behavior of DNN approximations to the conventional Sod shock tube solution to the compressible Euler equations of gas dynamics [56]. We use the shock tube

solution as a setting in which to explore what is meant by a DNN solution to a system of PDEs, what is meant by a regular (and irregular) solution to a system of PDEs both analytically as well as numerically, how these two frames of reference relate to each other, and where they differ. We spend some time examining what regular/irregular solutions really represent and how DNN solutions to differential equations provide an opportunity for transitioning into, what we refer to in this paper as a data-rational[1] paradigm where the critical and central focus of study turns to the content of the relational mapping between numerical systems and observation. Then we discuss in some detail how DNN methods compare and relate to more traditional and conventional ways of solving systems of PDEs.

The last portion of the paper is focused more directly on how to improve and extend the performance of DNN solutions in the context of irregular PDE solvers, and how to capitalize on the operational details of DNNs solvers to perform complete and simultaneous parameter space exploration. We also consider how to extend the framework of physics-based modeling to incorporate experimental data into the analytic workflow. These questions have growing importance in science and engineering applications as data becomes increasingly available. Many of the more traditional approaches, including data-informed parameter estimation methods [2] and real-time filtering [23], are frequently incorporated into more standard forward PDE solvers. We argue that DNN-based PDE solvers provide a natural interface between PDE solvers for physics-based models and advanced data analytics. The general observation is that because DNNs are able to combine a PDE solver framework with a simple and flexible interface for optimization, it becomes natural to ask how sufficient a modeling system is at representing experimental observations and to explore the nature of the mismatch between a simulated theoretical model and an experimentally measured phenomenon. In this context, the DNN framework automatically inherits the ability to couple many practical data-driven concepts, from signal analysis, to optimal control, to data-driven PDE enrichment and discovery.

To provide some insight into the utility of the DNN framework for solving PDEs, it is instructive to discuss some of the apparent strengths and drawbacks. Three strengths of the framework are:

(I) Phenomenal ease of prototyping PDEs,
(II) Natural incorporation of large data,
(III) Simultaneous solution over an entire parameter space.

Regarding (I), complicated systems of highly parameterized and multidimensional PDEs can be prototyped in TensorFlow or PyTorch in hundreds of lines of code, in a day or two. This might be compared to the decade-long development cycle of many legacy PDE solvers. For (II), incorporating and then utilizing experimental data in the PDE workflow as a straightforward supervised machine learning task is algorithmically very simple and provides a painless integration with the empirical scientific method. This feature could be naturally leveraged for practical uncertainty quantification, risk assessment, data-driven exploration, optimal control, and even discovery and identification of the theoretical underpinnings of physical systems (as discussed in some detail in section 4.4 below). In (III), another powerful and practical advantage is identified, where n-dimensional parameter space exploration simply requires augmenting the solution domain (space-time) with additional parameter axes, $(\boldsymbol{x}, t, p_1, \ldots, p_n)$, and then optimizing the DNN to solve the PDEs as a function of the parameters as inputs. The input space augmentation adds little algorithmic or computational complexity

[1] In this paper "data-rational" refers to a systems approach to differential equations that studies, as its primary focus, the mapping between the explicit numerical content of a differential system (e.g. a computer register) and the observed data (e.g. experimental measurements) it approximates. This topic is discussed in more detail in section 2.2.

over solving on the space-time domain $(\boldsymbol{x}, t)$ at a single parameter point $(p_1, \ldots, p_n)$ and is drastically simpler than exploring parameter space points sequentially.

Some of the present drawbacks of the DNN-based approach to solving PDEs include:

(i) Absence of theoretical convergence guarantees for the non-convex PDE residual minimization,
(ii) Slower overall run time per forward solve,
(iii) Weaker theoretical grounding of the method in standard PDE analysis.

We touch on these weaknesses throughout the paper, but briefly, (i) indicates the challenge posed by understanding convergence criteria in non-convex optimization, where solutions may get trapped in local minima. The concern of (ii) is substantially more subtle than it appears at first glance, and strongly depends on many aspects of the chosen neural network architecture, how optimized the hyperparameters are, what the ultimate goal of the simulation is, etc. Finally, (iii) merely highlights that DNNs have only recently been seriously considered for PDE solvers, and thus, remain largely theoretically untouched.

The paper is organized as follows: Section 2.3 provides a method overview of how DNNs can be used to solve a nonlinear system of PDEs. Within this section is a discussion about the relationship between different notions of solutions to PDEs and attempts to provide some context in which to understand where DNN solutions can be placed relative to solutions provided by more conventional numerical methods, and how they point towards 'data-rational' paradigms. Section 3 provides numerical experiments which demonstrate how the DNN solution behaves given unbounded gradients, how the DNN approximations of the Sod shock tube problem compare to more standard finite element and finite volume methods, and what is meant by concepts of convergence within this context. Finally section 4 expands on the standard DNN solution approach examining natural extensions to this framework. Here methods for improving convergence times of DNN solutions are discussed, as are solution dependencies on network architecture. The final subsection of section 4 discusses data-enrichment of PDEs, which is presented through the lens of a hypothetical experimental scenario, where the experimenter finds herself in the situation in which the anticipated physical model of the system does not adequately validate against the experimental data.

2. Background on Irregular Solutions and DNN Methods

In this section we first discuss some of the motivation underlying the use of DNN methods for solving differential equations, and how this naturally leads to a data-rational scientific paradigm. Next we review the DNN method for solving an n-coupled system of nonlinear PDEs, and then discuss how this method is directly applied to the shock problem.

2.1. Irregular Solutions to PDEs

Though many, if not most, known PDEs have been derived from and are studied for their ability to represent natural phenomena, at their core they remain mathematical abstractions that are only approximations to the phenomena that inspire them. These abstractions are often derived from simple variational perturbation theories, and are subsequently celebrated for their ability to capture and predict the physical behavior of complex natural systems. This "capturing" of physical systems is conventionally accomplished at the level of experimental validation, which is often many steps removed from the abstraction that the PDE itself represents.

In the study and understanding of PDEs and mathematical analysis, it is difficult to over-emphasize the importance of stability and regularity. One of the most challenging problems in modern theoretical science and engineering is developing a strategy for solving a PDE and ultimately interpreting the

solution. In this context, irregular solutions to PDEs are easy to come by. To account for this, convention introduces the concept of mathematically well-posed solutions (in the sense of Hadamard [24]). Well-posedness circumscribes what is meant, in some sense, by a "good" or admissible solution to a PDE. It is within this framework that the concept of classical and regular solutions, strong solutions, weak solutions, and so forth has evolved.

A "regular solution" can be thought of as one where the actual PDE is satisfied in an intuitively sensible and classical way. Namely, in a regular solution, enough derivatives exist at every point in the solution so that operations within the PDE make sense at every point in the entire domain of relevance. A well-posed regular solution is formally defined to provably exist, be unique, and depend continuously on the data given in the problem [15]. In contrast to regular solutions, the concept of a weak solution was developed to substantially weaken the notion of what is meant by a solution to a PDE. In weak solutions derivatives only exist in a "weak" distributional sense, and the solution is subsequently recast in terms of distributions and integral norms.

For example, in the case of a "strong solution" to a PDE, which is a "weak solution" that is bounded and stable in the L^2-norm and thus preserves important geometric properties, a solution may admit, e.g., pointwise discontinuities and remain an admissible solution. Such solutions are the strongest, and/or most well-behaved of the weak solutions; and yet still, interpreting the space of strong solutions from a physical point of view can be challenging. Moreover, weak solutions in the sense of Hadamard are only proper solutions to a PDE if they are adjoined with a concept of stability (such as L^p-stability) that implies a bound in a given normed space. This is a rather practical condition, in that a solution in a distributional sense that is not bounded in norm is too permissive to pointwise "variation" to be physically interpretable.

A substantial fraction of solutions arising in applied science are weak solutions that demonstrate stability in notably irregular spaces. Even the simplest of equations can admit solutions of highly irregular character (e.g., the Poisson equation [58]). In fact, in many applied areas the concept of turbulence becomes important, which from a mathematical point of view deals with PDEs in regimes where neither regularity nor stability are observed, a fact with broad conceptual ramifications [28]. This irregular and unexpected behavior in PDE analysis leads to a rather basic and inevitable question that we address in the next section: how well do our specific discrete numerical approximate solutions to certain PDEs capture their rigorously established mathematical properties, and why might this question matter?

2.2. Numerical solutions and the data-rational paradigm

Discrete numerical systems cannot, in general, exactly replicate infinite dimensional spaces (except in an abstract limit), and thus approximate numerical solutions to PDEs remain just that: approximations to mathematical PDEs. As a matter of practice, weak formulations are often implemented in numerical methods, but while these methods often preserve the spirit of their mathematical counterparts, they subsequently lose much of their content. For example, solutions in discontinuous finite element methods are frequently couched in terms of L^2 residuals and are presented along with numerical stability results, when in effect the discrete L^2 spaces they are purported to represent are simply piecewise continuous polynomial spaces that are capable of representing only a tiny fraction of admissible solutions in L^2.

It is then a natural question to ask, what has become of all the admissible solutions that even the most rigorous numerical methods discard by construction? And how exactly has it been determined which solutions are to be preserved? A pragmatist might respond with the argument that numerical methods develop, primarily, to deliver practical utility, adjoining as a central thematic pillar to their

evolution the notion of "physical relevance," where physical relevance is notionally defined in terms of the utility that the numerical solution provides in the elucidation of a practical observation.

While this may certainly be the case, and while we adopt the pragmatic perspective in this paper, we would like to raise a consideration to the reader: if numerical methods, particularly those driving the evaluation of predictive simulation models, are, with widespread application, utilizing numerical schemes that to some extent arbitrarily select solutions of a particular kind from a much larger space of mathematically admissible solutions, and then use that subset of solutions as the justification upon which interpretive predictive physical understanding is derived, then how can one evaluate the predictive capability of a model without directly comparing it to observed data?

As a consequence of these considerations, it seems more scientifically and logically responsible to operate under what might be characterized as a data-rational paradigm, that views a differential equation perhaps more primarily as a tool for approximating functional relationships between measureable and/or derivable quantities for the purposes of validating and predicting targeted systems. The numerical solution to a differential equation in this paradigm is more of a discrete relational mapping between an encoded numerical representation space (e.g. a computer register) and a network of observed data (e.g. experimental measurements). In this regard, the DNN solutions to differential equations explored in this paper may provide an opportunity for easy transition into such a data-rational paradigm. As DNN methods are often already couched within the context of optimization algorithms, data analytic frameworks and coding infrastructures (e.g. TensorFlow), it becomes natural and simple to incorporate and couple these frameworks with data-critical observational systems (as discussed in more detail in Section 4.4).

2.3. Method for solving PDEs using deep neural networks

In this paper we are interested in systems of n-coupled potentially nonlinear PDEs that can be written in terms of the initial-boundary value problem, for $i = 1, \ldots, n$:

$$\begin{aligned} \partial_t u_i + \mathcal{N}_i(\boldsymbol{u}; \boldsymbol{p}) &= 0, \quad \text{for } (\boldsymbol{x}, t, \boldsymbol{p}) \in \Omega \times [0, T_s] \times \Omega_{\boldsymbol{p}}, \\ u_i|_{t=0} &= u_{i,0}(\boldsymbol{x}, \boldsymbol{p}), \quad \text{for } (\boldsymbol{x}, \boldsymbol{p}) \in \Omega \times \Omega_{\boldsymbol{p}}, \\ \mathcal{B}_i(\boldsymbol{u}, \boldsymbol{p}) &= 0, \quad \text{for } (\boldsymbol{x}, t, \boldsymbol{p}) \in \partial\Omega \times [0, T_s] \times \Omega_p, \end{aligned} \tag{1}$$

over the domain $\Omega \subset \mathbb{R}^d$ with boundary $\partial\Omega$, and the m dimensional parameter domain $\boldsymbol{p} \in \Omega_{\boldsymbol{p}}$ given m parameters $\boldsymbol{p} = (p_1, \ldots, p_m)$, where $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{p})$ is the n dimensional solution vector with ith scalar-valued component $u_i = u_i(\boldsymbol{x}, t, \boldsymbol{p})$, T_s is the duration in time, $\mathcal{N}_i$ is generally a nonlinear differential operator, with arbitrary boundary conditions expressed through the component-wise operator $\mathcal{B}_i(\boldsymbol{u}, \boldsymbol{p})$.

Here we are interested in approximating the solutions to (1) using deep neural networks (DNNs). A feed-forward network can be described in terms of the input $y \in \mathbb{R}^{d_{\text{in}}}$ for $y = (\boldsymbol{x}, t, \boldsymbol{p})$, the output $z^L \in \mathbb{R}^{d_{\text{out}}}$, and an input-to-output mapping $y \mapsto z^L$, where d_{in} and d_{out} are the input and output dimension. In the present setting, $d_{\text{in}} = d + 1 + m$ and $d_{\text{out}} = n$. The components of the pre- and post-activations of hidden layer ℓ are denoted y_k^ℓ and z_j^ℓ, respectively, where the activation function is a sufficiently differentiable function $\phi : \mathbb{R} \to \mathbb{R}$. The jth component of the activations in the ℓth hidden layer of the network is given by

$$z_j^\ell(y) = b_j^\ell + \sum_{k=1}^{N_\ell} W_{jk}^\ell y_k^\ell(y), \quad y_k^\ell = \phi(z_k^{\ell-1}(y)), \tag{2}$$

where N_ℓ are the numbers of neurons in the hidden layers of the network, and W^ℓ_{jk} and b^ℓ_j are the weight and bias parameters of layer ℓ. These parameters make up the tunable, or "trainable", parameters of the network, which can be thought of as the degrees of freedom that parametrize the representation space of the DNN. Note that in section 4.3 we will consider a slightly more exotic network architecture, but for now, we use the standard "multilayer perceptron" described in (2).

In this context, the output of the neural network z^L, where the number of units N_ℓ is chosen and fixed for each layer, realizes a family of approximate solutions to (1), such that for each i component of (1),

$$z^L_i(\boldsymbol{x}, t; \boldsymbol{p}, \boldsymbol{\vartheta}) \approx u_i(\boldsymbol{x}, t, \boldsymbol{p}), \tag{3}$$

where we have set the parameters $\boldsymbol{\vartheta} = (W, b)$. Similarly, activation functions ϕ must be chosen such that the differential operators

$$\frac{\partial z^L_i}{\partial t}(\boldsymbol{x}, t; \boldsymbol{p}, \boldsymbol{\vartheta}) \quad \text{and} \quad \mathcal{N}_i(z^L; \boldsymbol{p}),$$

can be readily and robustly evaluated using reverse mode automatic differentiation [17].

We proceed by defining a slightly more general objective function to that presented in [51, 54],

$$J_i(\boldsymbol{\vartheta}) = J^{\text{PDE}}_i(\boldsymbol{\vartheta}) + J^{\text{BC}}_i(\boldsymbol{\vartheta}) + J^{\text{IC}}_i(\boldsymbol{\vartheta}) \tag{4}$$

where

$$\begin{gathered} J^{\text{PDE}}_i(\boldsymbol{\vartheta}) = \left\| \frac{\partial z^L_i}{\partial t} + \mathcal{N}_i(z^L; \boldsymbol{p}) \right\|_{\mathfrak{L}_i(\Omega \times [0,T_s] \times \Omega_{\boldsymbol{p}}, \wp_1)}, \quad J^{\text{BC}}_i(\boldsymbol{\vartheta}) = \|\mathcal{B}_i(\boldsymbol{u}, \boldsymbol{p})\|_{\mathfrak{L}_i(\partial\Omega \times [0,T_s] \times \Omega_{\boldsymbol{p}}, \wp_2)}, \\ J^{\text{IC}}_i(\boldsymbol{\vartheta}) = \|z^L_i(\boldsymbol{x}, 0, \boldsymbol{p}; \boldsymbol{\vartheta}) - u_{i,0}(\boldsymbol{x}, \boldsymbol{p})\|_{\mathfrak{L}_i(\Omega \times \Omega_{\boldsymbol{p}}, \wp_3)}, \end{gathered} \tag{5}$$

and $\mathfrak{L}_i$ indicates the form of the ith loss and $\wp_{1,2,3}$ denotes the probability density with respect to which the expected values of the loss are calculated. In particular, the form of the loss function can correspond to any standard norm (e.g., the discrete kernel of an L^p-norm), the mean square error, or specialized hybrids such as the Huber loss [27]. For this chosen form $\mathscr{L}$, the corresponding loss is given by the following expected value:

$$\|f(X)\|_{\mathfrak{L}(\mathcal{Y}, \wp)} = \mathbb{E}_\wp[\mathscr{L}(f)] = \int_{\mathcal{Y}} \mathscr{L}(f) \wp(X) dX.$$

Letting $z^L = \{z^L_1, \ldots, z^L_n\}$, we define any particular approximate solution $\hat{z}^L$ to (1) as finding a $\boldsymbol{\vartheta}$ that minimizes the loss relative to the parameter set. That is:

$$\hat{\boldsymbol{\vartheta}} = \arg\min_{\boldsymbol{\vartheta}} \sum_{i=1}^{n} J_i(\boldsymbol{\vartheta}). \tag{6}$$

To solve this optimization problem, one must approximate the loss function and couple this approximation with an optimization algorithm. For instance, using stochastic gradient descent (SGD), one uses a Monte Carlo approximation of the loss coupled with gradient descent. Thus, the loss is computed by drawing a set of samples X_{batch} from the joint distribution $\wp_1\wp_2\wp_3$ and forming the following approximation:

$$G_i(\boldsymbol{\vartheta}; X_{\text{batch}}) = G^{\text{PDE}}_i(\boldsymbol{\vartheta}; X_{\text{batch}}) + G^{\text{BC}}_i(\boldsymbol{\vartheta}; X_{\text{batch}}) + G^{\text{IC}}_i(\boldsymbol{\vartheta}; X_{\text{batch}}),$$

where

$$G_i^{\mathrm{PDE}}(\boldsymbol{\vartheta}; X_{\mathrm{batch}}) = \frac{1}{N_{\mathrm{PDE}}} \sum_{n=1}^{N_{\mathrm{PDE}}} \mathscr{L}(R_i(\mathbf{x}_n, t_n, \mathbf{p}_n)), \qquad G_i^{\mathrm{BC}}(\boldsymbol{\vartheta}; X_{\mathrm{batch}}) = \frac{1}{N_{\mathrm{BC}}} \sum_{n=1}^{N_{\mathrm{BC}}} \mathscr{L}(B_i(\mathbf{x}_n, \mathbf{p}_n)),$$

$$G_i^{\mathrm{IC}}(\boldsymbol{\vartheta}; X_{\mathrm{batch}}) = \frac{1}{N_{\mathrm{IC}}} \sum_{n=1}^{N_{\mathrm{IC}}} \mathscr{L}(z_i^L(\mathbf{x}_n, 0, \mathbf{p}_n; \boldsymbol{\vartheta}) - u_{i,0}(\mathbf{x}_n, \mathbf{p}_n)),$$

and $R_i(\mathbf{x}_n, t_n, \mathbf{p}_n)$ denotes the ith PDE residual evaluated at the batch point $(\mathbf{x}_n, t_n, \mathbf{p}_n)$, which is drawn according to $\wp_1$, and similarly for the BC and IC terms. Then, the SGD algorithm for the $k+1$ iteration can be written as

$$\boldsymbol{\vartheta}_{k+1} \leftarrow \boldsymbol{\vartheta}_k - \alpha_k \nabla_{\boldsymbol{\vartheta}} \left(\sum_{i=1}^{n} G_i(\boldsymbol{\vartheta}_k; X_{\mathrm{batch}}^k) \right), \tag{7}$$

where X_{batch}^k denotes the batch set drawn at iteration k.

In practice we find the Adam optimizer [30] to be the most efficient SGD-family algorithm for the cases run in this paper. For network initialization we use the Xavier uniform initializer [22]. It is also worth noting that the hyperparameters L and N_ℓ as well as the sampling distribution $\wp$ are additional hyperparameters that can be optimized. For simplicity, however, we do not perform hyperparameter optimization in this paper.

2.3.1. Shocks

Perhaps the simplest type of irregularity that confronts a numerical method is a numerical shock. A numerical shock need not be a mathematical discontinuity, and as such it is a fundamentally different concept than both a mathematical shock and a physical shock. One definition of a numerical shock is: the numerical response (in the form of numerical instability) of a method when the representation space that the method is constructed in does not support the function space it is interrogating. Examples of this behavior can be found from Gibb's phenomena in spectral methods to Runge's phenomena in polynomial based methods and elsewhere. More broadly, one can simply posit that the radius of convergence of the solution must be contained within the support of its discrete representation, or else instabilities are likely to arise [12, 41]. As a consequence, numerical approaches to the shock problem display unique characteristics that often expose deep numerical insights into the nature of the method. The unique signature that a numerical method displays when dispersing its instabilities throughout a solution foretells the predictive characteristics of the simulation itself.

Many successful numerical methods can be viewed as variations of general strategies with important but subtle differences in the underlying detail. Consider, for example, finite volume methods (FVM), spectral, pseudospectral, and Fourier Galerkin methods, continuous and discontinuous Galerkin finite element methods, etc. In each of these cases, the system of PDEs is cast into a weak formulation, where the spatial derivatives are effectively transferred to analytically robust basis functions, and fluxes are recovered from integration by parts. For example, if we choose some adequately smooth test function $\boldsymbol{\varphi}$, multiply it by some transport equation $\partial_t \boldsymbol{U} + \boldsymbol{F}(\boldsymbol{U})_x = 0$ in a state space where $\boldsymbol{U}$ a state vector, and integrate by parts, we have an equation that can be written in the form:

$$\frac{d}{dt} \int_\Omega \boldsymbol{\varphi} \odot \boldsymbol{U} dx + \int_{\partial\Omega} \boldsymbol{\varphi} \odot \boldsymbol{F} dS - \int_\Omega \boldsymbol{F} \odot \boldsymbol{\varphi}_x dx = 0, \tag{8}$$

where $\odot$ denotes componentwise multiplication.

To see the relation between distinct numerical methods, recall here that when $\boldsymbol{\varphi}$ is a high order polynomial, and the discrete domain is, for example, a single element, the recovered method is a type of spectral element method (SEM). Whereas, when the discretization is turned into a finite element mesh Ω_h comprised of i subintervals, $\Omega_1, \ldots, \Omega_i$, and the basis functions are chosen to be continuous or piecewise discontinuous polynomials, we have the continuous Galerkin (CG) or discontinuous Galerkin (DG) finite element methods, respectively. Similarly, when the elements Ω_i are viewed as finite volumes and the basis functions $\boldsymbol{\varphi}$ are chosen to be piecewise constant, a finite volume method can be easily recovered.

As discussed above the weak formulation (8) of a method should not really be viewed, in and of itself, as an advantage in the data-rational paradigm. Indeed, the DNN loss functions (4) could equally be cast into a weak formulation (as in [7] for example), paying a price in the simplicity of the method. Similarly, by using different activation functions (e.g., ϕ being taken as ReLU), the universal approximation theorem can, in the limit sense, recover all Lebesgue integral function spaces [38]. We sidestep these issues, viewing them as unnecessary complications that muddy the simplicity and elegance of the DNN solution. Instead, we choose the hyperbolic tangent $\tanh(z)$ activation function and interpret solutions to shock problems through the lens that smooth solutions are dense in L^p.

2.3.2. Euler's equations and the Sod shock tube

For simplicity, we start by considering the dimensionless form of the compressible Euler equations in 1D, solved over only the space-time domain $\Omega \times [0, T_s]$, with $\Omega = [-1, 1]$ and boundary $\partial\Omega = \{-1, 1\}$. We are interested in solutions to the initial-boundary value problem:

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = 0, \quad \boldsymbol{U}_{|t=0} = \boldsymbol{U}_0, \quad \boldsymbol{U}_{|x\in\partial\Omega} = \boldsymbol{U}_{\partial\Omega}, \tag{9}$$

for a state vector $\boldsymbol{U} = (\rho, \rho u, E)^\top$ and corresponding flux $\boldsymbol{F}(\boldsymbol{U}) = (\rho u, \rho u^2 + p, u(p+E))^\top$. The state vector is comprised of the density ρ, velocity u, and total energy of the gas

$$E = \frac{p}{\gamma - 1} + \frac{\rho}{2} u^2,$$

while the pressure-temperature relation is the single component ideal gas law $p = \rho T$. The constant γ denotes the adiabatic index and, unless otherwise noted, is taken throughout the remainder of the paper to be $\gamma = 5/3$, while the speed of sound is $c_s = \sqrt{\gamma T}$.

To analyze the behavior of the network on a simple and classic 1D shock problem, we solve the Sod shock tube. The initial condition is chosen to be piecewise constant:

$$\boldsymbol{U}_0 = \begin{cases} \boldsymbol{U}_l, & x < 0 \\ \boldsymbol{U}_r, & x \geq 0 \end{cases} \tag{10}$$

where the left state is defined by $\boldsymbol{U}_l = (1, 0, 1.5)^\top$ and the right by $\boldsymbol{U}_r = (0.125, 0, 0.15)^\top$. This also defines the left and right Dirichlet boundaries $\boldsymbol{U}_{\partial\Omega} = \{\boldsymbol{U}_l, \boldsymbol{U}_r\}$. The exact solution to this system is readily obtained by a standard textbook procedure [35]. Except for in section 4.4, the simulation time horizon is set to $T_s = 0.2$.

3. Numerical Experiments, Results, and Discussion

In this section we examine the basic numerical behavior of DNN solutions to the Sod shock tube, present some results and discussion. In section 3.1 we show how the DNN solution behaves without any regularization and discuss how that compares to other numerical methods. In section 3.2 we discuss how to add analytic diffusion to regularize the solution. Section 3.3 compares the regularized DNN results against those from standard finite volume and discontinuous Galerkin finite element methods.

3.1. An unlimited solution using DNNs

The standard solution to the Sod shock tube problem for (9) poses real challenges to most numerical methods. In the standard DG method, for example, when the polynomial degree is nonzero, $p > 0$, the solution becomes numerically unstable without the use of some form of slope limiting [44, 43]. Similarly when FVM is evaluated at higher than first order accuracy, the absence of flux-limiting leads to unstable solutions [59, 14]. This behavior is also observed in spectral methods [60, 21], continuous Galerkin methods [39], and so on.

One of the more robust regimes for solving shock problems can be found in constant DG methods with degree $p = 0$ basis functions (or equivalently, FVM methods with first order accuracy), wherein both a stable and relatively robust solution can be readily calculated, but at the cost of large numerical diffusion. Indeed the ability to solve these types of irregular problems helps explain the preference practitioners often demonstrate in choosing low order methods to compute solutions to irregular systems, where to resolve solution features one relies solely on mesh/grid refinement.

In contrast to lower order approaches, the DNN solution to the Sod shock tube problem is able to exploit the relative flexibility of DNN functional approximators, as determined by composition of affine transformations and element-wise activation nonlinearities. For example, with hyperbolic tangent activations, the DNN solution to the shock tube problem becomes an approximation to a regular solution of the Sod problem at high order accuracy, in the sense that the approximation order of a linear combination of transcendental functions is C^∞.

In the case of the DNN solution, even though the discontinuous shock fronts display unbounded derivatives in the classical sense, the DNN is able to find a relatively well-behaved solution as shown in Fig. 1. The DNN formulation with the hyperbolic tangent activation, if the optimization is to converge to a parameter point $\hat{\boldsymbol{\vartheta}}$, is predicated on a regular solution. Failure to converge could arise if the derivatives at the locus of discontinuity, and the components of $\boldsymbol{\vartheta}$, enter unstable unbounded growth in the iterations of the optimization protocol. Indeed, the DNN approximates the discontinuous mathematical shock precisely in the limit in which certain components of $\boldsymbol{\vartheta}$ tend to infinity. We observe that in practice, however, this does not happen. The inherent spectral bias [49] of the DNNs seems to be able to contain the potential instability of the solution until the optimization has settled in a "reasonable" local minimum. This leads to an interesting feature of the DNN solutions, namely that they seem to demonstrate robustness while broadly maintaining many of the small-scale features of the exact solution, even in the presence of irregularity in the formulation.

3.1.1. DNNs as gridless adaptive inverse PDE solvers

To explain why an approximate regular DNN solution to a shock problem can perform as well as it does, as shown in Fig. 1, we note that a DNN solver can really be viewed as a type of gridless adaptive inverse solver. First, the absence of a fixed grid in the evaluation of the loss functions (4) and determination of the optimization direction (7) means that the probability density $\wp$ can be finitely sampled in space as densely as the method can computationally afford at every iteration and for as

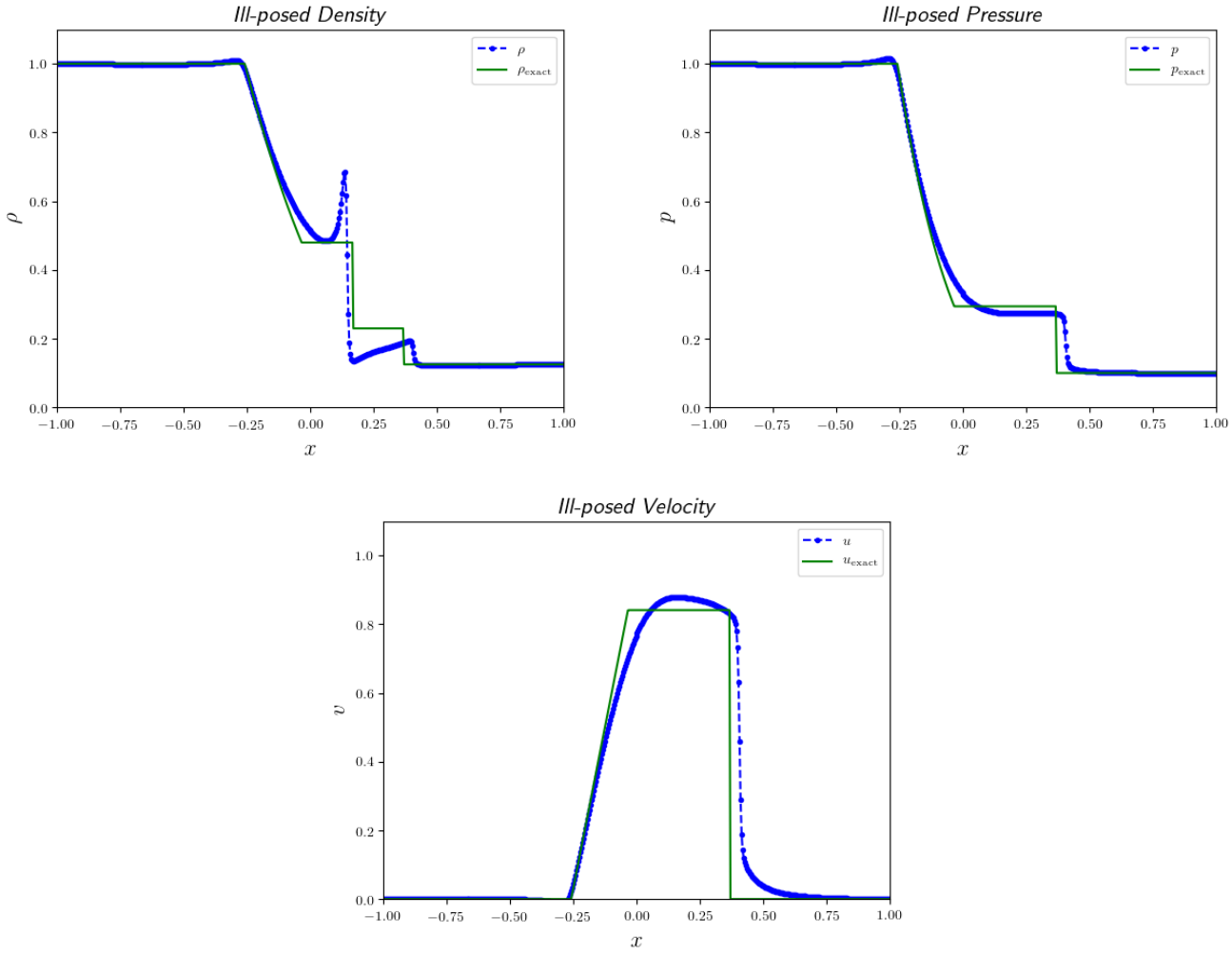

Figure 1: The unlimited solutions to the Sod shock tube. The solution is fully ill-posed in the classical formulation, leading to undefined behavior (i.e., unbounded derivatives) along the shock fronts. Nevertheless, the solution remains stable (in contrast with the unlimited versions of other algorithms), and appears relatively robust to the major features of the solution.

many iterations as can be afforded. Given the computational resources, this can lead to a relatively densely sampled interrogation of the space-time domain. Moreover, adaptive mesh methods for shock problems have long been known to be highly effective [6, 34], particularly when shock fronts can be precisely tracked, and there are a discrete number of them. However, AMR and *hp*-adaptive methods are constrained by their meshing/griding geometries, even as isoparametric and isogeometric methods have been developed to, at least in part, reduce these dependencies [42].

The second pertinent observation about the DNN solution is that because it utilizes an optimization method to arrive at the solution, it can be viewed as a type of inverse problem. Unlike a forward numerical method, which accumulates temporal and spatial numerical error as it runs, the DNN solutions become more accurate with iterations, being a global-in-space-time inverse-like solver. This means that it becomes entirely reasonable to consider running such systems in single or even half precision floating point formats, leading to potential performance benefits in comparison to more standard PDE solvers. As an example, all DNN computations done in this paper are run in single precision.

3.2. Diffusing along unbounded gradients

As discussed above, solving a shock problem without adding some form of diffusion (often in the form of limiters or low order accurate methods with upwind fluxes) tends to lead to unstable solutions in classical numerical methods. Similarly here, even in the case of the DNN solution, when searching for an approximation to the regular solution shown in Fig. 1, the resulting solution, though reasonably well-behaved, is of course spurious along the undefined shock fronts.

As in all numerical methods, to ameliorate this problem, all that needs to be done is to add some small amount of diffusion to constrain the derivatives along the shock fronts. In this case, we slightly alter (9) into the non-conductive compressible Navier-Stokes type system,

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) - \partial_x \boldsymbol{G}(\boldsymbol{U}) = 0, \quad \boldsymbol{U}_{|t=0} = \boldsymbol{U}_0, \quad \boldsymbol{U}_{|x\in\partial\Omega} = \boldsymbol{V}_{\partial\Omega}, \tag{11}$$

given a dissipative flux $\boldsymbol{G}(\boldsymbol{U}) = (0, \rho\tau u_x, \frac{1}{2}\rho\tau u_x^2)^\top$. Again here we start by merely solving over the space-time domain, $\Omega \times [0, T_s]$, given a fixed and positive constant $\tau \in \mathbb{R}^+$. As we show in more detail below, adding a modest amount of viscosity $\tau = 0.005$ (i.e. Reynolds number $\sim$340) leads to a DNN solution that is competitive with some of the most effective numerical methods available for solving the Sod shock tube problem.

3.3. Numerical benchmarks

In this section, we show numerical results obtained for the DNN with artificial viscosity and compare to analogous results obtained with DG finite element methods (FEMs) and FVMs for the Sod shock tube problem. Such a comparison is fraught with complexity for at least two reasons. First, the iterative convergence of the DNN approach is different from that of a typical method, since it is posed as an optimization problem rather than a nonlinear system of equations. Because the optimization problem is non-convex, we must accept that the optimal DNN approximation is not necessarily unique, and, even if it is, may converge instead to a local minimum. However, as long as the loss function at this local minimum is small, the resulting DNN represents a reasonable approximation. Fig. 3 shows a typical convergence result as a function of iteration number for the DNN method, which is analogous to the convergence behavior of SGD type problems in general. Note here that all runs in this paper are run to losses of comparable magnitude, i.e. RMSE between $\sim 2 \times 10^{-4}$ and $\sim 8 \times 10^{-5}$.

Second, it is not clear how to make the comparison between the DNN approach and typical methods fair. While total time to solution at a target accuracy is likely the most practically relevant metric, the time to solution for the DNN method depends on a long list of choices and tunable parameters that affect the optimization algorithm, including, for example, the batch size and the learning rate. We have made little attempt to optimize these parameters in this work, and thus, we choose not to compare based on time to solution.

Instead, we use proxies for the complexity of the DNN. In comparing typical discretization schemes, it is common to make comparisons at constant numbers of degrees of freedom (DoFs), essentially using the DoFs as a proxy for time to solution. In the context of the DNN, there are at least two quantities that affect time to solution in a qualitatively similar way to DoFs in a standard method: the number of batch points, which is somewhat analogous to grid density or number of quadrature points in a typical method, and the number of tunable parameters (i.e., DoFs) in the DNN, which is analogous to the number of DoFs in a standard method. Thus, we compare the methods, both graphically and using the mean squared error, in terms of the spatial "grid" and the number of DoFs.

3.3.1. Solutions as a function of spatial grid density

A way to evaluate the relationship between the DNN method and FVM or DGFEM is by comparing the relative density of effective residual evaluation points in space. Broadly, in the DG solution, for degree $p = 1$ polynomial solutions, this corresponds to $(p + 1)N_{\text{el}}$ for N_{el} the number of elements. In the FVM method this corresponds to simply N_{vol}, the number of finite volumes. In the DNN the residual is evaluated at batch points determined by the sampling distribution $\boldsymbol{x} \sim \wp$, which we denote with N_{batch}.

Solution Type	Density	Pressure	Velocity
Unlimited DNN	0.00199	0.00074	0.01272
FVM 1st Order	0.00117	0.00139	0.00804
FVM 2nd Order	0.00030	0.00019	0.00164
DG, $p = 2$	0.00163	0.00259	0.01413
DG, $p = 5$	0.00314	0.00811	0.02945
DNN, $\tau = .005$	0.00020	0.00025	0.00136

Table 1: Mean square error of the unlimited DNN solution in Fig. 1 and the solutions as a function of the grid density in Fig. 2, against the exact Sod shock tube solution.

One nuance in comparing the methods through residual evaluation points is that in standard FVM and DGFEM methods, these points are spatially fixed. In the DNN approach however, the batch points are resampled after every evaluation of the objective function gradients. As such, it might seem more appropriate to look at how DNNs compare to *hp*-adaptive DGFEM and FVM that use adaptive mesh refinement (AMR). Note, however, that adaptive forward problems using DGFEM and FVM are adaptive in a fundamentally different way relative to the residual evaluation as compared to the DNN method. In forward problems, the residual evaluation shifts in space relative to the current solution in time. In the DNN solution, however, though the batch points are resampled, the samples can be chosen independent of the solution behavior, and are done so over the whole spacetime domain $\Omega_h \times [0, T_s]$.

This dependence on space and time of the sampled batch points N_{batch}, distinguishes the DNN solution from even the adaptive FVM and FEM solutions. Thus in order to compare the spatial resolution of the two different types of solutions, we compare solutions using the rule of thumb that $\sqrt{N_{\text{batch}}} \sim N_{\text{vol}} \sim (n+1)N_{\text{el}}$, as these capture the average relative 'grid density' along one dimension.

Results of this comparison are shown in Fig. 2 and Table 1, where the timestepping is chosen sufficiently small. Here the DGFEM solution is run at $p = 2$, $N_{\text{el}} = 24$ and at $p = 5$, $N_{\text{el}} = 12$ using a Rusanov Riemann solver with adaptive hierarchical slope limiting [44]. Both first and second order FVM solutions are run at $N_{\text{vol}} = 72$, where the first order solution is unlimited, and the second order uses a standard per-face slope limiter. The DNN solution is run at $\sqrt{N_{\text{batch}}} = 72$, using $\tau = 0.005$. The results indicate that the DNN solution performs favorably to the FVM and DGFEM solutions relative to spatial grid density.

3.3.2. Solutions as a function of degrees of freedom

Another way of comparing the DNN solution to the FVM and DGFEM methods is to look at solutions as a function of their respective degrees of freedom (DoFs). Heuristically this can be thought of as comparing the number of "tunable parameters" within the representation space of the solution.

Since the solution in the DNN is a global in space and time solution, we consider the DoFs spanning the whole space, $\Omega_h \times [0, T_s]$. In the DG solution this corresponds to $S(n + 1)N_{\text{el}}T_{\text{grid}}$, where T_{grid}

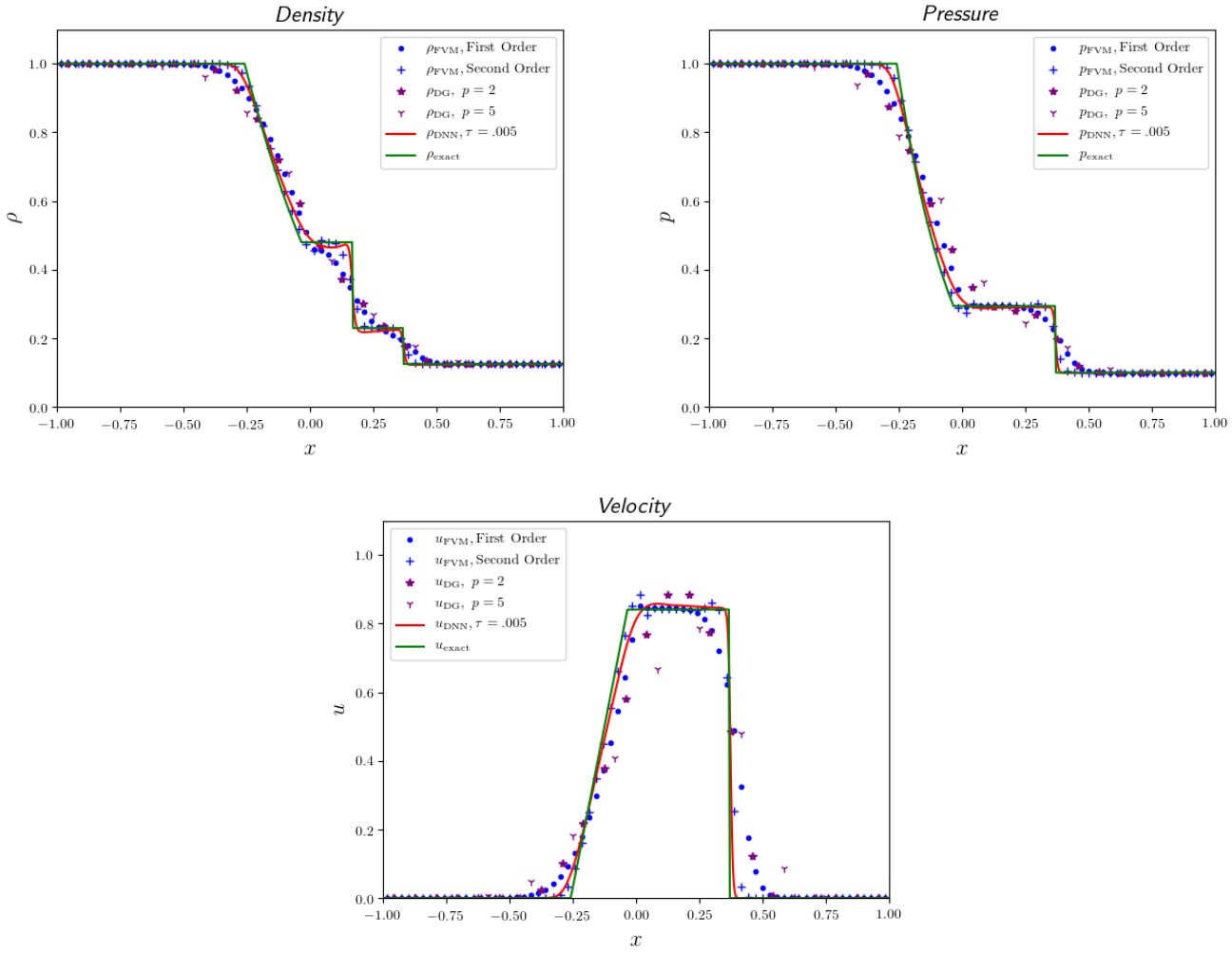

Figure 2: Comparison of slope-limited discontinuous Galerkin (DG) finite element method solvers, flux-limited finite volume method (FVM) solvers, and the dissipative DNN solver holding fixed the relative resolution in the spatial grids of each solution, $\sqrt{N_{\text{batch}}} \sim N_{\text{vol}} \sim (n+1)N_{\text{el}}$.

is the temporal grid and S are the number of stages in the Runge-Kutta discretization. Likewise in the FVM setting, we have $SN_{\text{vol}}T_{\text{grid}}$. In the case of the DNN, on the other hand, the degrees of freedom are recovered by counting the scalar components of the vector $\boldsymbol{\vartheta}$ parametrizing the network architecture (2), and this yields

$$\text{DNN}_{\text{DoFs}} = N_1(d_{\text{in}}+1) + d_{\text{out}}(N_L+1) + \sum_{\ell=1}^{L-1} N_{\ell+1}(N_\ell+1), \tag{12}$$

where L is the number of hidden layers, N_ℓ is the width of the ℓ-th layer, and d_{in} and d_{out} are the input and output dimensions, respectively.

The results are presented in Fig. 4 and Table 2. In the DNN solution, the degrees of freedom are $\text{DNN}_{\text{DoFs}} = 248065$ after setting $L = 16$ and $N_\ell = 128$. The DGFEM solutions are run the same as in section 3.3.1, except that at sixth order, $p = 5$, $S = 4$, $N_{\text{el}} = 104$ and $T_{\text{grid}} = 100$, so that $\text{DGFEM}^{(6)}_{\text{DoFs}} = 249600$. Similarly the third order accurate DGFEM solutions use $p = 2$,

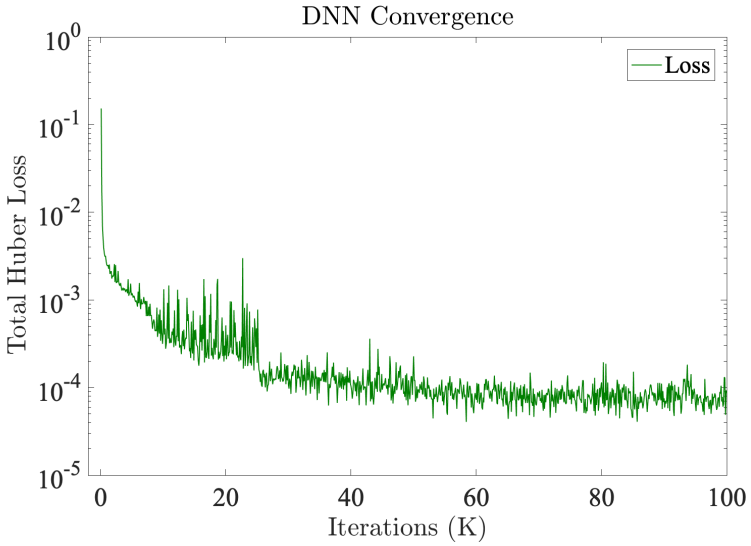

Figure 3: Here we show the convergence behavior on the problem in Fig. 2, using a decreasing learning rate schedule of $\times 0.1$ every 25K iterations, starting at 10^{-4}.

$S = 4$, $N_{\text{el}} = 208$ and $T_{\text{grid}} = 100$, corresponding to $\text{DGFEM}^{(3)}_{\text{DoFs}} = 249600$. The first order accurate FVM solution is run the same as in section 3.3.1, except $S = 1$ (a forward Euler solver is used), $N_{\text{vol}} = 500$, and $T_{\text{grid}} = 500$ so that $\text{FVM}^{(1)}_{\text{DoFs}} = 250000$. At second order the FVM method uses a predictor-corrector method, which we set as $S = 2$, $N_{\text{vol}} = 350$, and $T_{\text{grid}} = 355$ corresponding to $\text{FVM}^{(2)}_{\text{DoFs}} = 248500$. Again, the results show that the DNN solution compares favorably as a function of degrees of freedom.

Solution Type	Density	Pressure	Velocity
FVM 1st Order	0.00027	0.00017	0.00136
FVM 2nd Order	0.00011	0.00005	0.00084
DG, $p = 2$	0.00021	0.00013	0.00182
DG, $p = 5$	0.00042	0.00029	0.00257
DNN, $\tau = .005$	0.00020	0.00025	0.00136

Table 2: Mean square errors of the solutions as a function of fixed degrees of freedom in both space and time, as shown in Fig. 4

4. Natural Extensions

The remainder of the paper is dedicated to examining natural extensions to the DNN framework applied to the Sod Shock tube problem. The first of these is an application of a type of "simulated annealing," that can be used along the viscous and/or dissipative parameter directions to improve the effective time to convergence in the method. Next we show how solving the DNN framework along a full parameter axis, in this case along the adiabatic index axis γ, can lead to a simultaneous and fully parameterized solution in the entire parameter space at nearly no additional cost. We then examine the behavior of an LSTM-like network for this system. And finally we discuss what might be the most practical application of the DNN methods for shocktube type problems, which is: PDE solutions via data-enrichment.

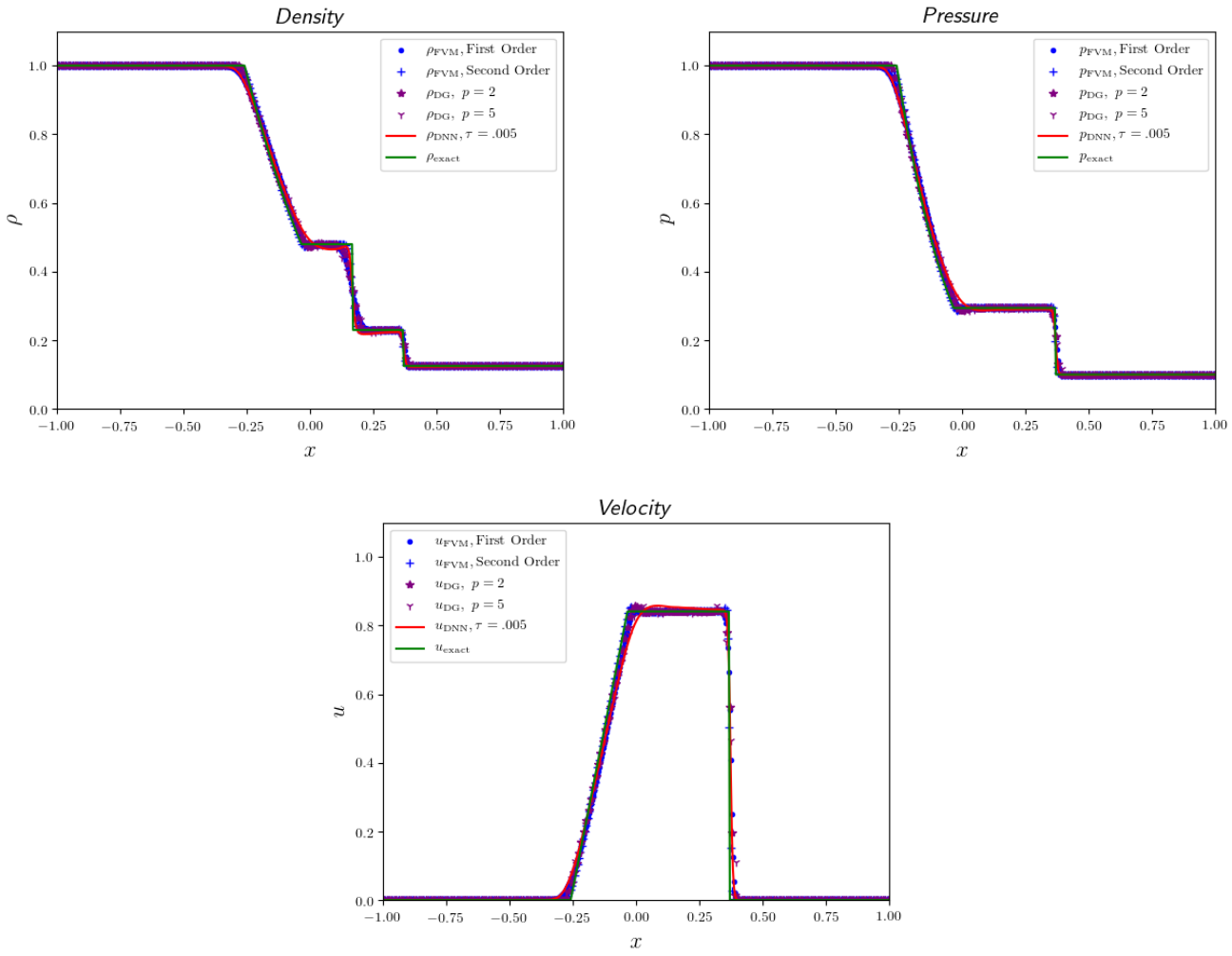

Figure 4: Comparison of slope-limited discontinuous Galerkin (DG) finite element method solvers, flux-limited finite volume method (FVM) solvers, and the dissipative DNN solver holding fixed the total degrees of freedom (in both space and time) of the discrete representation.

4.1. Dissipative Annealing

One of the potential limitations of the DNN approach, in comparison to other numerical methods, is the compute time-to-convergence. Discussion of convergence in the DNN/PDE setting is complicated by the fact that "convergence" refers to different concepts in numerical PDE analysis and in numerical optimization. In numerical PDE analysis, convergence usually refers to rate at which an approximate solution tends to the exact solution as a function of the representation parameters, e.g., mesh width h and/or polyniomial order p. In iterative numerical optimization, however, convergence refers to the rate at which the parameters $\boldsymbol{\vartheta}$ converge to a (possibly local) minimum as a function of the number of iterations. The global minimum may exist only in the limit in which some components of $\boldsymbol{\vartheta}$ tend to infinity. Then, heuristics mandate that we seek convergence to an acceptable local minimum. It is thus difficult to arrive at a formal definition for what is meant by time-to-convergence in the DNN setting, and this can be easily seen in Fig. 3. In this section, we resort to a hybrid heuristic between the two concepts of convergence discussed above, and refer to "time-to-convergence" as the number of computational cycles required to reach a comparable level of accuracy to that of a more traditional PDE solution method. It is in this sense that DNN solutions can appear noticeably more expensive,

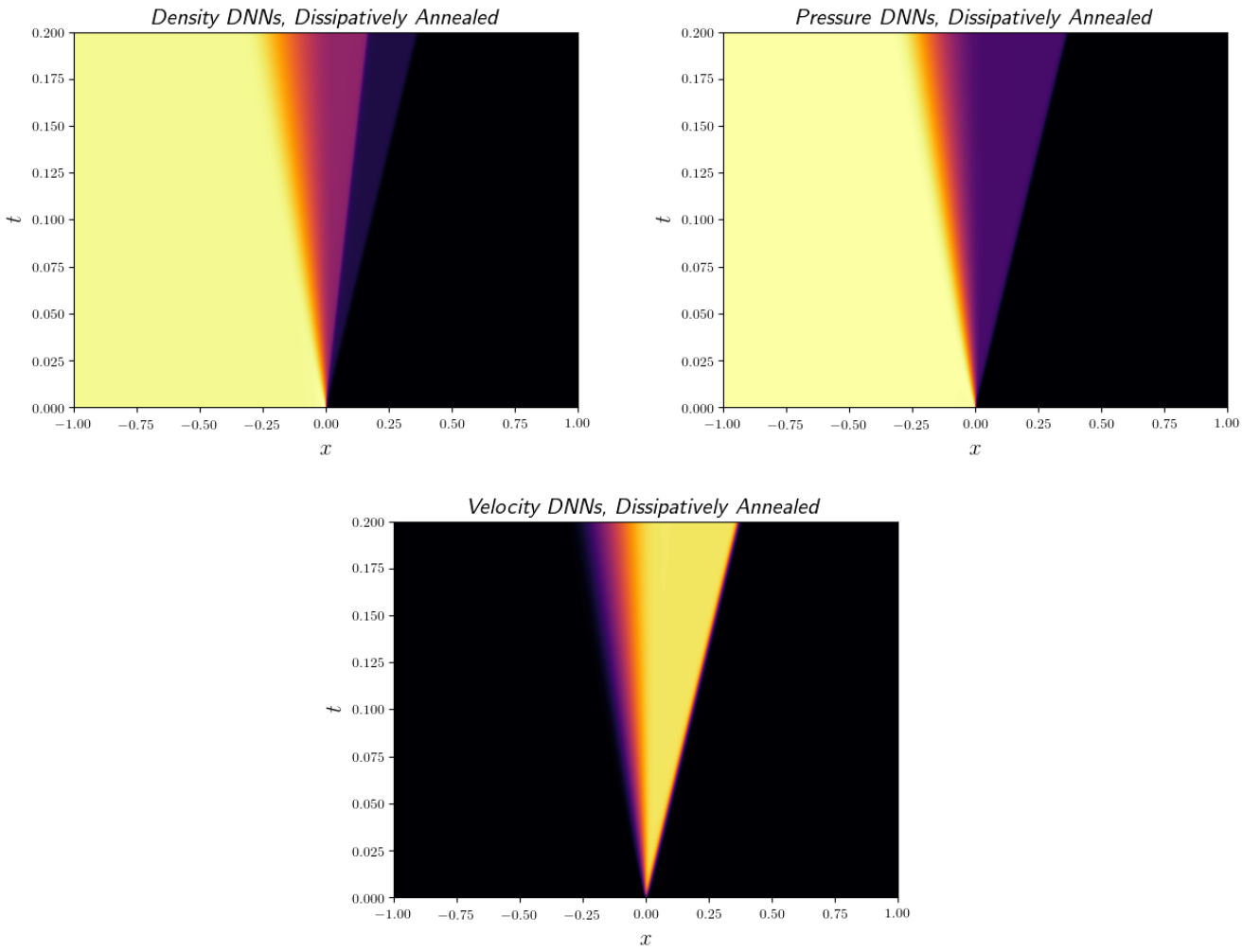

Figure 5: The Sod shock tube solution using a DNN solver along with dissipative annealing, out to only 60K iterations.

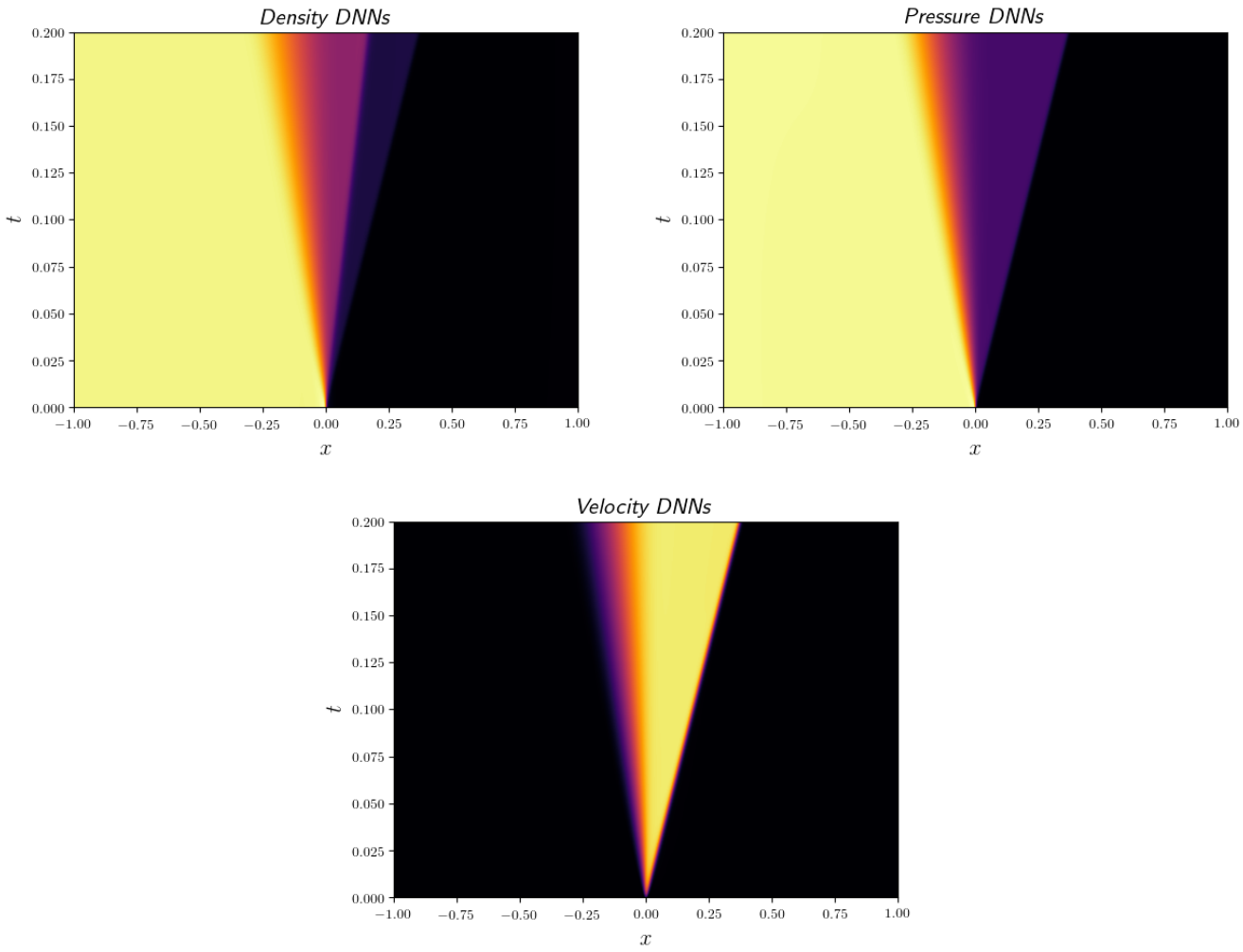

Figure 6: The Sod shock tube solution using a DNN solver *without* dissipative annealing to 100K iterations.

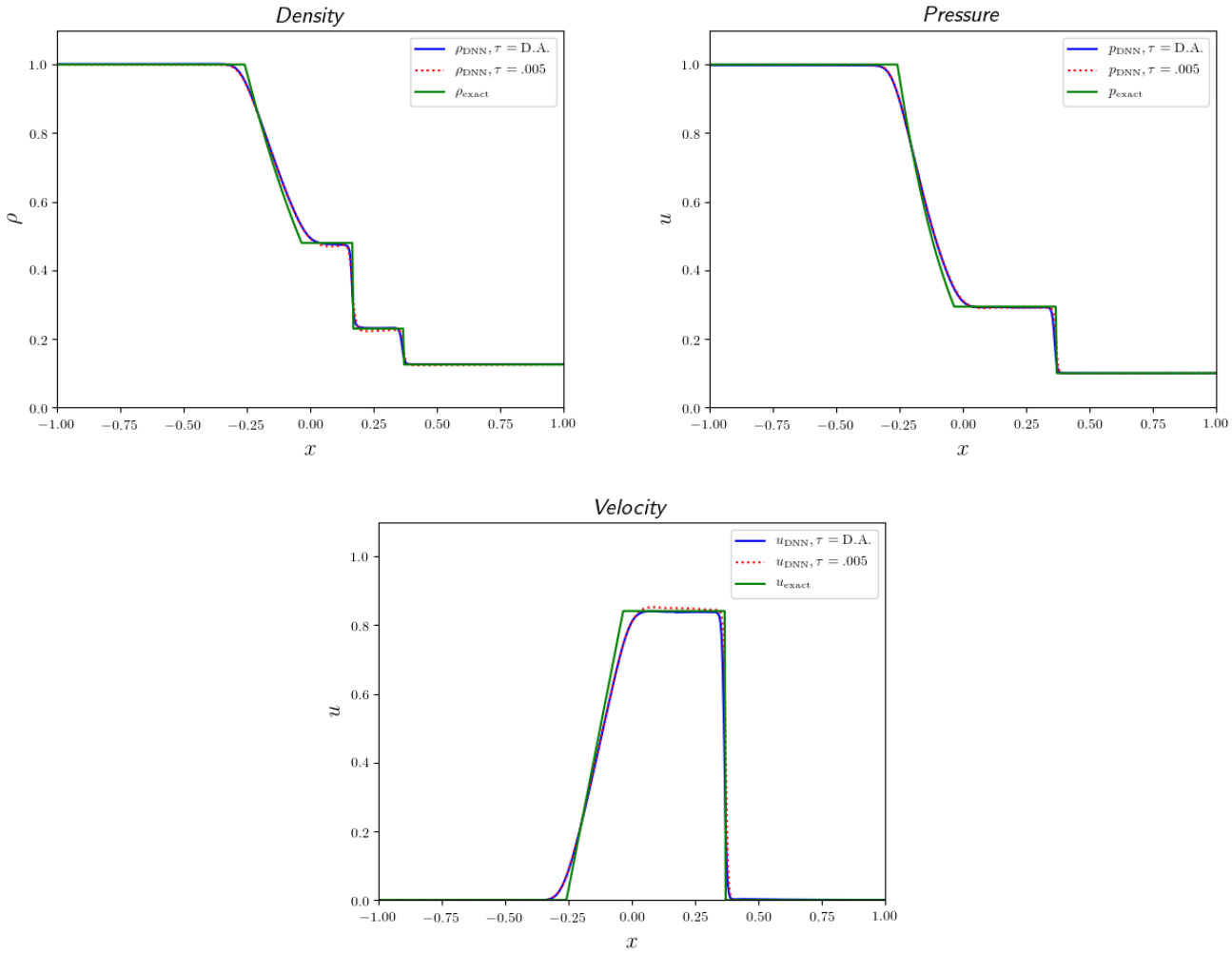

Figure 7: The final timestep of Sod shock tube solution comparing the dissipative annealed solution at 60K, to the standard solution at 100K iterations, to the exact solution.

though in this section we discuss some of the nuances that underlie this observation, and why the unique capacity and flexibility of DNN solutions makes the time-to-convergence less restrictive than it seems.

The premise we assume here about slowly converging solutions (in the sense of optimization) is that large-scale features of a smooth solution are relatively easy to converge to, but when accompanied with finer scale features, the optimization noise from attempting to fit the finer scales can obscure the larger scale features from the optimizer. To mitigate this effect, inspired by the simulated annealing method [31], we recast the dissipative flux in (11) as

$$\boldsymbol{G}(\boldsymbol{U}) = (0, \rho\tau_0 u_x, \tfrac{1}{2}\rho\tau_0 u_x^2)^\top,$$

where $\tau_0 = \tau_0(l)$ becomes an iteration-l numerical viscosity. In this case $\tau_0(l) = g(l)\tau_{\text{smooth}}$ for some smooth $\tau_{\text{smooth}} \in \mathbb{R}$. The function $g(l)$ is chosen as a fractional stepwise function bounded from above by unity.

The idea of using τ_0, is to converge early and fast in the iteration cycle to an overtly smooth solution of (11) with large viscosity. Since such a solution has dramatically fewer fine-scale features, it is conjectured that the optimizer can more easily find a stable minimum of the objective function for such a smooth solution. As a consequence, fewer total iterations are required to converge to the minimum.

We have tested this idea on the Sod problem, and have found that with minimal effort it seems to reduce the number of iterations needed to arrive at similar results. For example, in Figs. 5–7, we test a DNN using a decreasing learning rate schedule. This base case is run with the minibatch size of 5000 and initial learning rate of 10^{-4} that is reduced every 25000 iterations by factor of 0.1 for a total of 100000 iterations. This case is compared to a dissipatively annealed solution obtained with the same minibatch size, but instead of 25000 iterations per learning rate decrement we find we can get away with 12000 iterations, where after each boundary set, $g = \{1, 0.2, 0.18, 0.1386, 0.1\}$, such that $\tau_0(l) = g(l)\tau_{\text{smooth}}$ for $\tau_{\text{smooth}} = 0.05$ and $l = 1, \ldots, 5$. This means the dissipatively annealed solution takes only 60000 iterations to arrive at $\tau_0(5) = 0.005$.

As is clear from Figs. 5–7, the results are nearly indistinguishable. This result, however, should be taken with a grain of salt. In this example case the two runs are set up exactly the same, up to the dissipative anealing algorithm. It is not clear that this is an entirely fair comparison, given the number of hyperparameters that can be tuned in each case, and provided the non-deterministic nature of SGD.

4.2. Simultaneous parameter scans

Dense parameter space exploration is the most reliable way to develop predictive input-output mappings for scenario development, optimal design and control, risk assessment, uncertainty quantification, etc., in many real-world applications. Without understanding the response to the parameters of the system, be they variable boundary conditions, interior shaping constraints, or constitutive coefficients, it is all but impossible to examine the utility an engineered model might have in some particular design process. PDEs clearly help in this regard since they are fully parametrized models of anticipated physical responses. That being said, PDEs are also often fairly expensive to run forward simulations on, frequently requiring large HPC machines to simulate even modest systems [8, 40]. Consequently, parameter space exploration of a PDE system is usually both very important and very computationally expensive.

To mitigate the expense of running forward models, surrogate modeling [16] and multifidelity hierarchies [48] are often conceived in order to develop cost-effective ways of examining the response

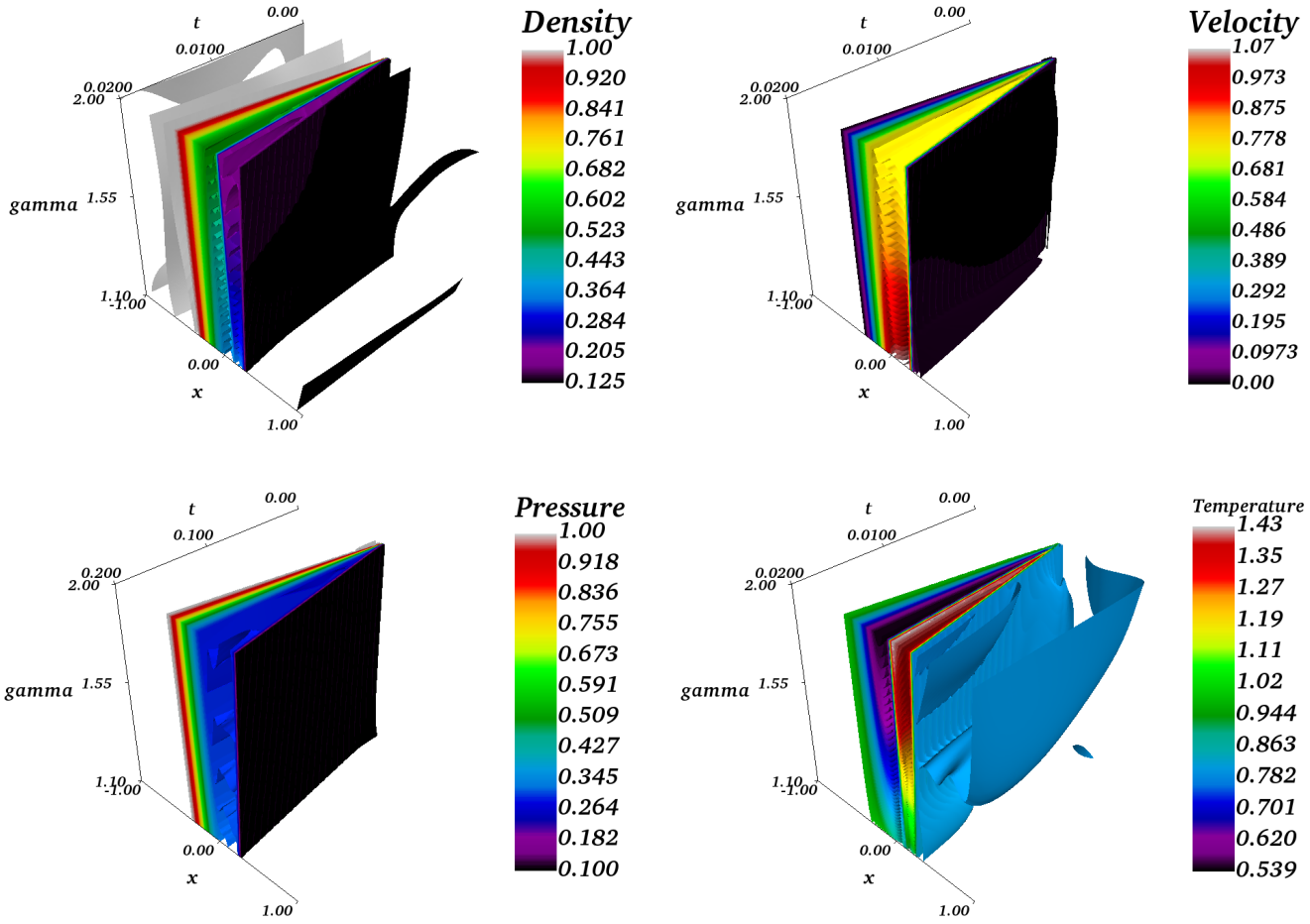

Figure 8: 100 3D contours of the Sod shock tube solution solved over the continuous parameter scan, $(x, t, \gamma) \in [-1, 1] \times [0, 0.2] \times [1.1, 2.0]$ at 100K iterations.

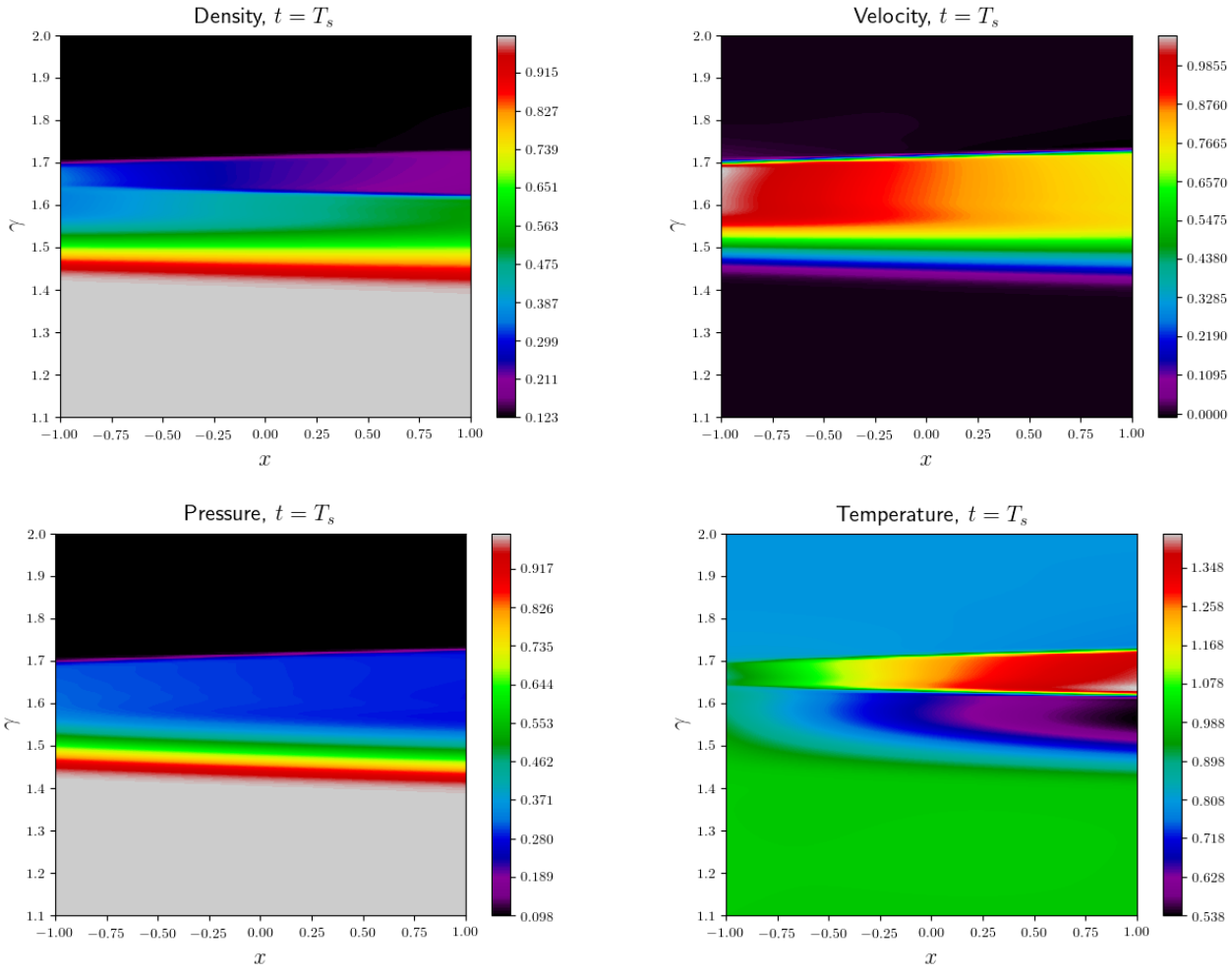

Figure 9: Variation in γ and x of the Sod shock tube solution, at $T_s = 0.2$ with 100K iterations.

of a PDE relative to its parametric responses. One of the ways in which these methods function is by developing reduced models to emulate the parametric dependencies from a relatively sparse sampling of the space. In this way it becomes possible—for a sufficiently smooth response surface—to tractably parametrize the input-output mapping, and thus interrogate the response at a significantly reduced cost, lending itself to statistical inference, uncertainty quantification, and real-world validation studies.

However, as is commonly the case, the response surface is not sufficiently smooth, or its correlation scale-length cannot be determined a priori, and these reduced order mappings can become highly inaccurate. In such circumstances having a way to emulate the exact response surface would clearly be of high practical value. As a potential solution to this, one of the most powerful immediate features of the DNN framework seems to be the ability to perform simultaneous and dense parameter space scans with little effort. The framework is thus a natural, and in some sense exact emulator for complicated system response models. In the case of the Sod shock problem this can be accomplished by simply recasting (11) relative to its parameter γ, where instead of solving over space-time $(x,t) \in \Omega \times [0,T_s]$, the solution is solved over the parameter-augmented parameter space $(x,t,\gamma) \in \Omega \times [0,T_s] \times [\gamma_{\text{low}}, \gamma_{\text{high}}]$.

Intuitively one might expect the increase of dimensionality to be prohibitive, as the parameter space grows from a discrete 2D system to a discrete 3D system. However, in analogy with the observation of section 4.1 where the parameter τ is dissipatively annealed, this is not what is observed. Instead, using the same DNN parameter sets, the same minibatch size and learning rate schedule, optimization over the parameter-augmented input space (x,t,γ) reaches similar loss values with almost no computational overhead. Contour and time slice plots of the solutions are shown in Fig. 8 and Fig. 9 for the density, velocity pressure, and temperature, each as a function of $(x,t,\gamma) \in [-1,1] \times [0,0.2] \times [1.1,2.0]$.

Remarkably, in this example, the DNN framework demonstrates that it is as cost effective to perform dense parameter space exploration along the dimension of the adiabatic index γ as it is to solve at a single fixed value of γ. It remains unclear how robust this behavior is over physical parameters of the model — in this case (11). But however robust this behavior ends up being, even if only across isolated and specific parameters, this demonstrates a remarkably advantageous aspect of the DNN formulation.

4.3. DNNs with residual connections and multiplicative gates

As pointed out in [54], the DNN architecture has a significant influence on its solution behavior. In [54] it is shown that an LSTM-inspired feed-forward network architecture with residual connections and multiplicative gates can help improve performance for some parabolic PDE problems. These models are effectively an alternative to the more standard deep neural network architecture presented in (2).

We implement these LSTM-like architectures for our 1D mixed hyperbolic-parabolic like PDE system (9), to see how it behaves relative to standard DNNs in the context of more irregularity in the solution space. Setting $\boldsymbol{\zeta} = (\boldsymbol{x}, t)$, we test the Euler system (9) on the following architecture

comprised of $L+1$ hidden layers:

$$\begin{aligned}
S^0 &= \phi(W^0\boldsymbol{\zeta} + b^0),\\
Z^\ell &= \sigma(U^{z,\ell}\boldsymbol{\zeta} + W^{z,\ell}S^\ell + b^{z,\ell}), \quad \ell = 0,\dots,L-1,\\
G^\ell &= \sigma(U^{g,\ell}\boldsymbol{\zeta} + W^{g,\ell}S^\ell + b^{g,\ell}), \quad \ell = 0,\dots,L-1,\\
R^\ell &= \phi(U^{r,\ell}\boldsymbol{\zeta} + W^{r,\ell}S^\ell + b^{r,\ell}), \quad \ell = 0,\dots,L-1,\\
H^\ell &= \phi(U^{h,\ell}\boldsymbol{\zeta} + W^{h,\ell}(S^\ell \odot R^\ell + b^{h,\ell}), \quad \ell = 0,\dots,L-1,\\
S^{\ell+1} &= (1-G^\ell)\odot H^\ell + Z^\ell \odot S^\ell, \quad \ell = 0,\dots,L-1,\\
z^L(\boldsymbol{\zeta},\boldsymbol{\theta}) &= W^L S^L + b^L,
\end{aligned} \tag{13}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function and the network parameters are given by:

$$\begin{aligned}
\boldsymbol{\theta} = \Big\{ & W^0, b^0, \left(U^{z,\ell}, W^{z,\ell}, b^{z,\ell}\right)_{\ell=0}^{L-1},\\
& \left(U^{g,\ell}, W^{g,\ell}, b^{g,\ell}\right)_{\ell=0}^{L-1}, \left(U^{r,\ell}, W^{r,\ell}, b^{r,\ell}\right)_{\ell=0}^{L-1}, \left(U^{h,\ell}, W^{h,\ell}, b^{h,\ell}\right)_{\ell=0}^{L-1}, W^L, b^L \Big\},
\end{aligned} \tag{14}$$

and the number of units per layer is N_ℓ. For more details on the network architecture we refer the reader to [54].

To understand whether the standard DNN architecture is more effective, in some sense, than (13), it is essential to recognize that the degrees of freedom scale differently for the LSTM-like system (13) than they do in (12), where the LSTM-like system is substantially more expensive per network layer ℓ. More explicitly, assuming that each layer has the same width, N_ℓ, the degrees of freedom in the LSTM-like system can be calculated as:

$$\mathrm{LSTM}_{\mathrm{DoFs}} = 4LN_\ell^2 + (4L(d_{\mathrm{in}}+1) + d_{\mathrm{in}} + d_{\mathrm{out}} + 1)N_\ell + d_{\mathrm{out}}. \tag{15}$$

Comparing (12) to (15), for a fixed number of layers $L = 2$, setting for (9) $d_{\mathrm{in}} = 2$ and $d_{\mathrm{out}} = 3$, the resulting relationship between the number of units in the LSTM, $N_\ell \implies N_{\mathrm{LSTM}}$, and DNN, $N_\ell \implies N_{\mathrm{DNN}}$, can be computed by solving the following ceiling function:

$$N_{\mathrm{DNN}} = \left\lceil \frac{-7 + \sqrt{16N_{\mathrm{LSTM}}^2 + 72N_{\mathrm{LSTM}} + 49}}{2} \right\rceil.$$

For our numerical test, we set $L = 16$ with the reference solution width set to $N_{\mathrm{DNN}} = 128$. This leads to $\mathrm{DNN}_{\mathrm{DoFs}} = 248451$. Solving for the relationship above this yields $N_{\mathrm{LSTM}} = 63$. The results are presented in Fig. 10, and show fairly unambiguously that for this particular test case the LSTM-like architecture does not perform as well as the standard DNN. As above, this result must be taken in context. It may well be, for example, that the tuned parameters for standard DNNs do not tune the LSTM-like networks well, and that the result presented is not a proper comparison. Again, to fully reveal the relationship between these architectures on even just this one simple hyperbolic test case, would require a full study of its own.

That said, from a purely practical point of view, the implementation of the DNN versus the implementation of the LSTM-like networks seems to lead to a network architecture with a large disparity in the number of graph edges. The DNN network has effectively one layer per $\ell \in L$, while

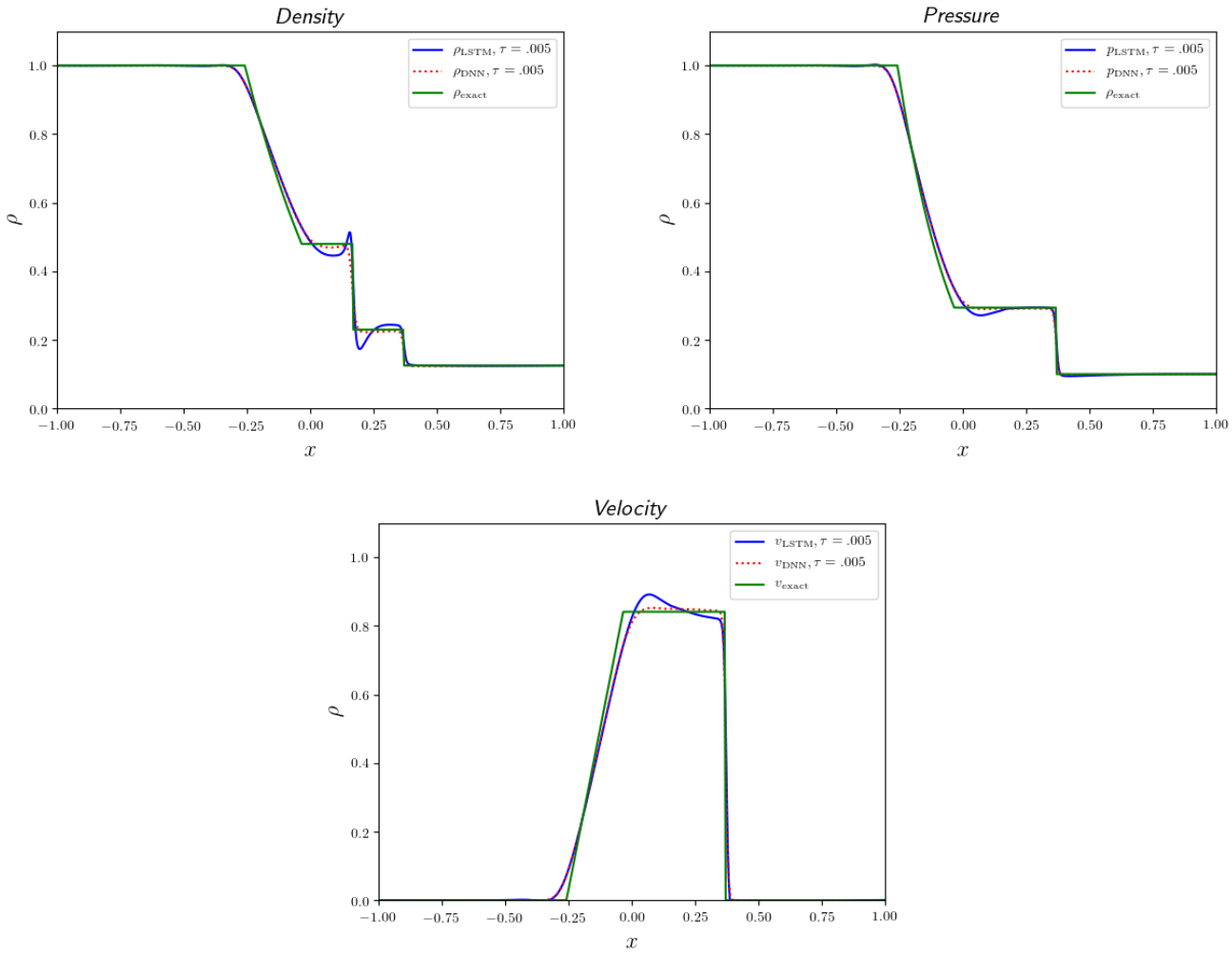

Figure 10: The Sod shock tube solution at the final timestep, comparing the LSTM-like architecture versus standard DNN at fixed DoFs.

the LSTM-like network has, in some sense, five layer-like networks per $\ell \in L$ as seen in (13. The result of this is that in our implementation, even at fixed degrees of freedom, the LSTM-like network takes almost five times longer in compute time to finish than the standard DNN, at a fixed number of iterations. Again, it is not clear if this slowdown can be reduced by using a more clever implementation of the LSTM-like networks, or if this is simply a result of increasing the network complexity in the graph.

4.4. Data-enriched PDEs

The simplicity and apparent robustness of DNNs for solving systems of nonlinear and irregular differential equations raises the question: how can one incorporate experimental data into such a framework? Work combining DNN-based approaches for solving PDEs with conventional supervised machine learning has already started to emerge, and we presume that this trend will only accelerate. For example, Raissi et al. [51] introduced "data discovered PDEs," where a relatively traditional parameter estimation is performed over differential operators that are discovered through optimization. This parameter estimation can be performed for a $\lambda \in \mathbb{R}$ that factors through a differential operator $\lambda u u_x$. This can be expanded to data driven discovery of PDEs, where parameter hierarchies can be used to generate libraries of operators that are selected based on system specific physical criteria and constraints, in order to effectively "construct" and/or discover PDE systems from large data sets [37]. These types of approaches are emerging with increasing interest [52, 53, 5, 9, 50], and naturally lend themselves to the DNN frameworks.

These types of empirical PDE discovery techniques offer clear advantages in cases where the systems are assumed to be too complex to easily construct first principle models. However, researchers also find themselves in a different type of situation, where the physics model that describes the system response is confidently prescribed, so that any mismatch between the model and data raises questions about the experimental setup just as it raises them about the model.

To explore this common circumstance, we address a situation relating to the experimental validation of a first principles model system. First principles models are predicated on the idea that the physical dependencies within an experimental system are, or can be, fully understood. Validation studies, however, frequently discover that this is in fact not the case [20, 47]. Frequently systematic behaviors, engineering details, alleatoric and/or epistemic uncertainties can cascade, and lead to systems with behaviors that are uncertain and/or different from those of the model systems anticipated to describe them.

In circumstances such as these, a researcher might be in a situation where a prescribed model system does not plausibly validate against the large quantities of high resolution experimentally measured data that have been collected. Some of the scientific questions one might want to raise in such circumstances are:

1. Are my experimental results reliable?
2. Is my model system sufficient to describe the experimental data?
3. How far from my model system is the measured data?
4. What is the form of the mismatch between the data and the model?
5. How might I enrich my model system to more accurately capture the observed behavior?
6. Are there characteristics in the mismatch that reveal missing physical subsystems?

Below we present a simple example of how to approach such a circumstance, and provide an outline of how such a series of questions can be systematically retired.

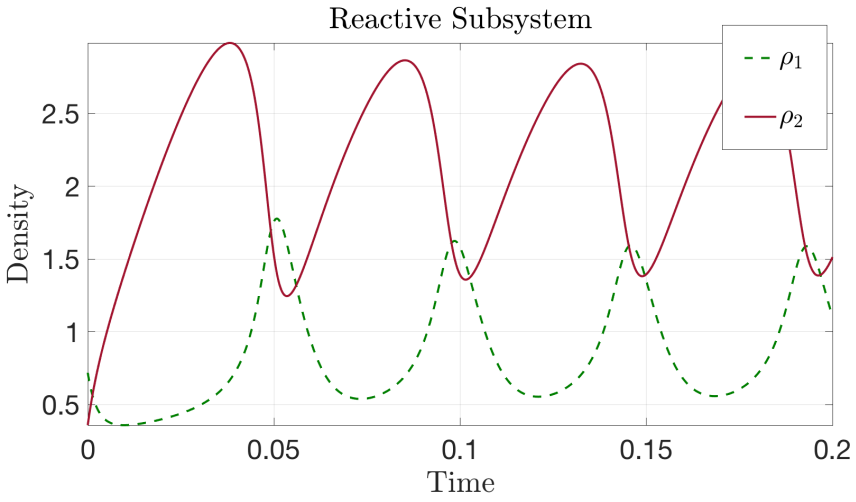

Figure 11: The reactive subsystem from (18). To illustrate the autocatalytic oscillation, the initial state is set to $\rho_1 = 2\rho_l/3, \rho_2 = \rho_l/3$, though the autocatalysis is largely independent of the initial conditions.

4.4.1. Hypothetical experimental senario

We consider the following hypothetical scenario: an experimental test facility is set up to run experiments interrogating the behavior of transonic gas dynamics. All solutions in this section are run using the DNN method for solving PDEs. Although the experimental setup is mildly exotic, the expected responses of the system are anticipated to obey classical gas dynamics (9). Experiments are run, and a process is initially set up to validate relative to the following dimensionless ideal model system:

$$\begin{aligned} \partial_t \rho + (\rho u)_x &= 0, \\ \partial_t (\rho u) + \left(\rho u^2 + p - \tau \rho u_x\right)_x &= 0, \\ \partial_t E + (Eu + pu - \tau \rho u u_x - \kappa T_x)_x &= 0, \end{aligned} \tag{16}$$

where the total energy density is given by

$$E = \frac{p}{\gamma - 1} + \frac{\rho}{2} u^2,$$

and the initial-boundary data is set to the same as in (9). Here a constant heat conductivity $\kappa = \tau = 0.005$ is also assumed. In the process of validating the system, the classical gas dynamic model (16) repeatedly demonstrates that it does not strongly validate against the experimentally measured data. After making sure the measurement equipment is well calibrated, and the confidence in the data is high, the researchers are left with the quandary: what is happening?

A traditional approach to this problem might begin with a vigorous debate between the simulation experts and the laboratory experts, both claiming systematic errors on behalf of the other. Both camps might proceed by testing their subsystems to the best of their ability and coming to the conclusion that nothing is wrong, per se, but there is some other physics occurring in the test facility that they have not properly anticipated. As a consequence, the next step might be to slowly start adding back physics terms to (16). Subsequent efforts may include parameter estimation on remaining uncertain parameters in the model until a better fit can be recovered. In this case, however, such a process would be extremely slow and tedious as illustrated below.

Unbeknownst to both the modellers and experimentalists, the actual dynamics going on inside the

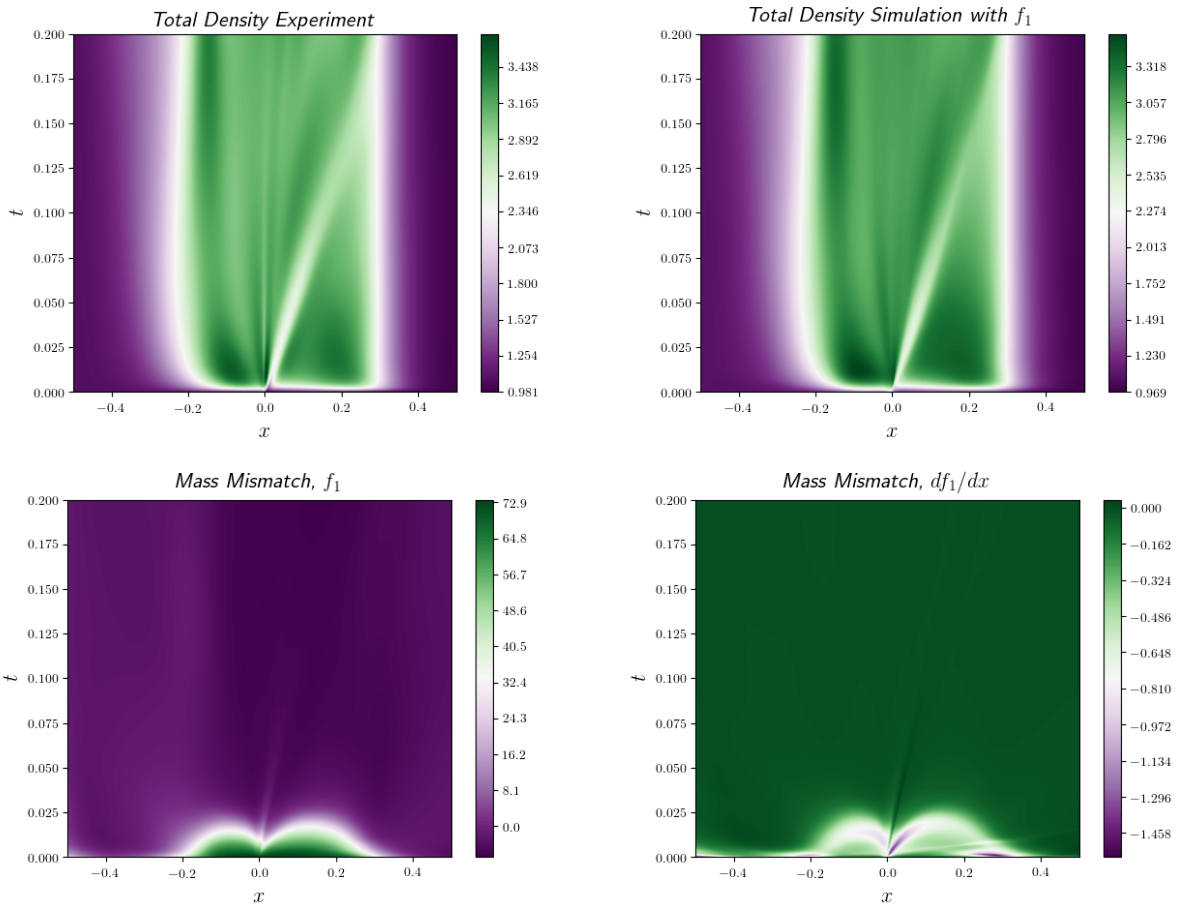

Figure 12: The top left shows the result from the total measured density in the experiment. The top right is the DNN-enriched simulated value of the density after adding the f_i's. The bottom panels show the function f_1 and it's first derivative, where its dominant signature effect can be found near the initial state of the system, $t = 0$.

Table 3: The initial-boundary data for the model (16) and the experiment (19).

B.C.	ρ	u	v	w	T	B_x	B_y	B_z
Left Mod. B.C.	1.08	1.2	–	–	0.8796	–	–	–
Right Mod. B.C.	0.9891	-0.0131	–	–	0.9823	–	–	–
Left Exp. B.C.	1.08	1.2	0.01	0.5	0.8796	2.0	3.6	2.0
Right Exp. B.C.	0.9891	-0.0131	0.0269	0.010037	0.9823	2.0	4.0244	2.0026

experiment, is a substantially more complicated system. In fact, it turns out that a feature of the experimental setup is inadvertaintly inducing ionization of the gas in the chamber. As a result, the gas actually behaves like a reactive photoactivated autocatalytic plasma in the center of the reactor, characterized exactly by the following equations:

$$
\begin{aligned}
&\partial_t \rho_1 + (\rho_1 u)_x = (k_1 A_1 + k_2 \rho_1^2 \rho_2 - k_3 \rho_1 A_2 - k_4 \rho_1)\zeta,\\
&\partial_t \rho_2 + (\rho_2 u)_x = (k_3 \rho_1 A_2 - k_2 \rho_1^2 \rho_2)\zeta,\\
&\partial_t(\rho u) + \left(\rho u^2 + p + \frac{1}{8\pi}\boldsymbol{B}^2 - \frac{1}{4\pi}B_x^2 - \tau\rho u_x\right)_x = 0,\\
&\partial_t(\rho v) + \left(\rho uv - \frac{1}{4\pi}B_x B_y - \tau\rho v_x\right)_x = 0,\\
&\partial_t(\rho w) + \left(\rho uw - \frac{1}{4\pi}B_x B_z - \tau\rho w_x\right)_x = 0,\\
&\partial_t E + \left(Eu + pu + \frac{1}{8\pi}\boldsymbol{B}^2 - \frac{1}{4\pi}B_x(\boldsymbol{v}\cdot\boldsymbol{B}) - \tau\rho u u_x - \kappa T_x\right)_x = 0,\\
&\partial_t B_x = 0, \quad \nabla\cdot\boldsymbol{B} = 0,\\
&\partial_t B_y + (uB_y - vB_x)_x = 0,\\
&\partial_t B_z + (uB_z - wB_x)_x = 0,
\end{aligned}
\tag{17}
$$

where

$$
E = \frac{p}{\gamma - 1} + \frac{\rho}{2}\boldsymbol{v}^2 + \frac{1}{8\pi}\boldsymbol{B}^2, \quad \text{and} \quad \sum_i \rho_i = \rho.
$$

Here $\boldsymbol{B} = (B_x, B_y, B_z)$ is a magnetic field, thought inconsequential in the erroneously presumed absence of ions, and the velocity field in the gas is defined as $\boldsymbol{v} = (u, v, w)$. In this hypothetical scenario, just like for the model system (16), the initial conditions for all unknowns are initialized with a jump discontinuity at $x = 0$, over the domain $\Omega = [-0.5, 0.5]$, for $t \in [0, 0.1]$, as inspired in the RP2 case in [13]. The remaining full boundary conditions are listed in Table 3.

The autocatalytic reactive subsystem is induced by virtue of an unexpected ionization pulse in the reactor, leading to an oscillating chaotic attractor characterized by the reactions:

$$
\begin{aligned}
A_1 &\xrightarrow{k_1} \rho_{\mathbb{1}}\\
2\rho_{\mathbb{1}} + \rho_2 &\xrightarrow{k_2} 3\rho_{\mathbb{1}}\\
A_2 + \rho_{\mathbb{1}} &\xrightarrow{k_3} \rho_2 + A_3\\
\rho_{\mathbb{1}} &\xrightarrow{k_4} A_4
\end{aligned}
\tag{18}
$$

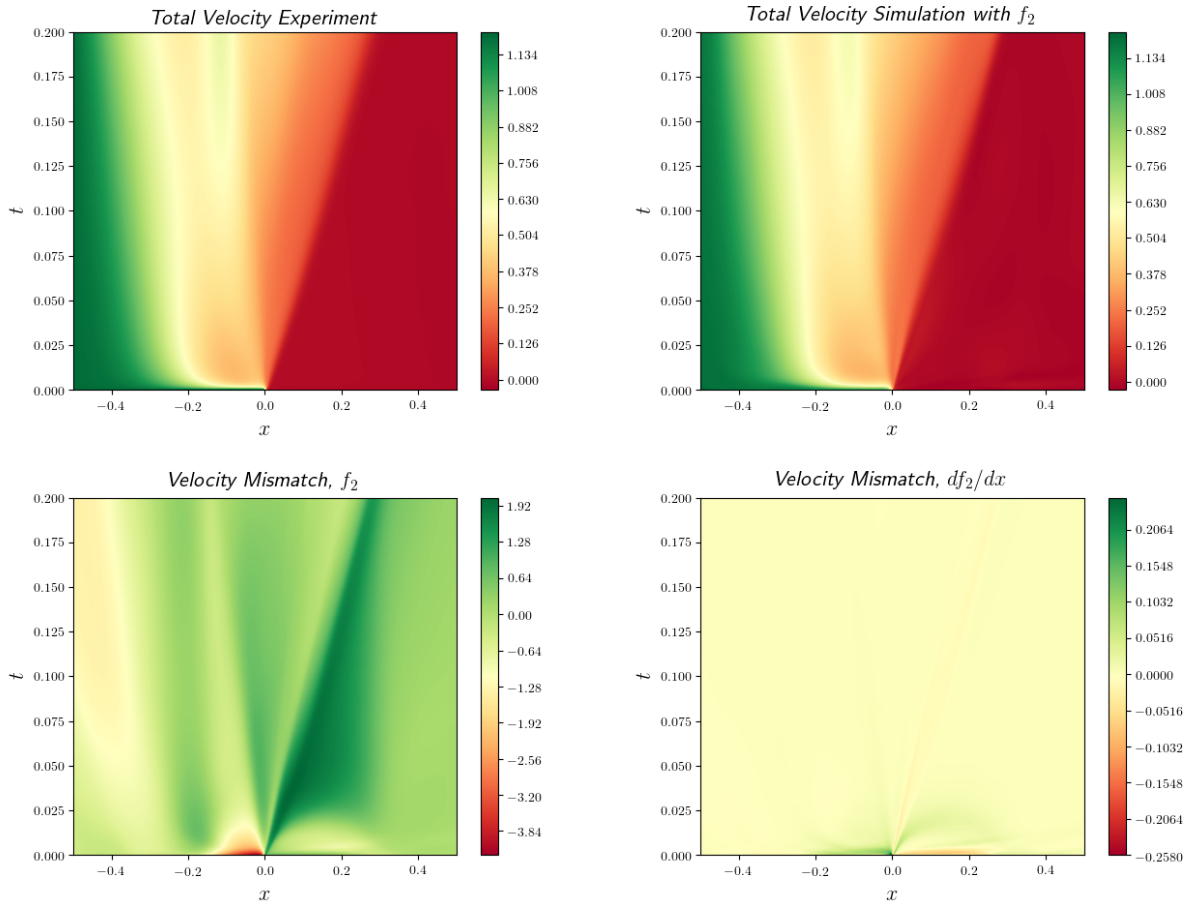

Figure 13: The top left is the experimentally measured velocity, and the top right the DNN-enriched simulation including the contribution from the mismatch function f_2. The mismatch function f_2 is shown on the bottom left, with df_2/dx on the bottom right.

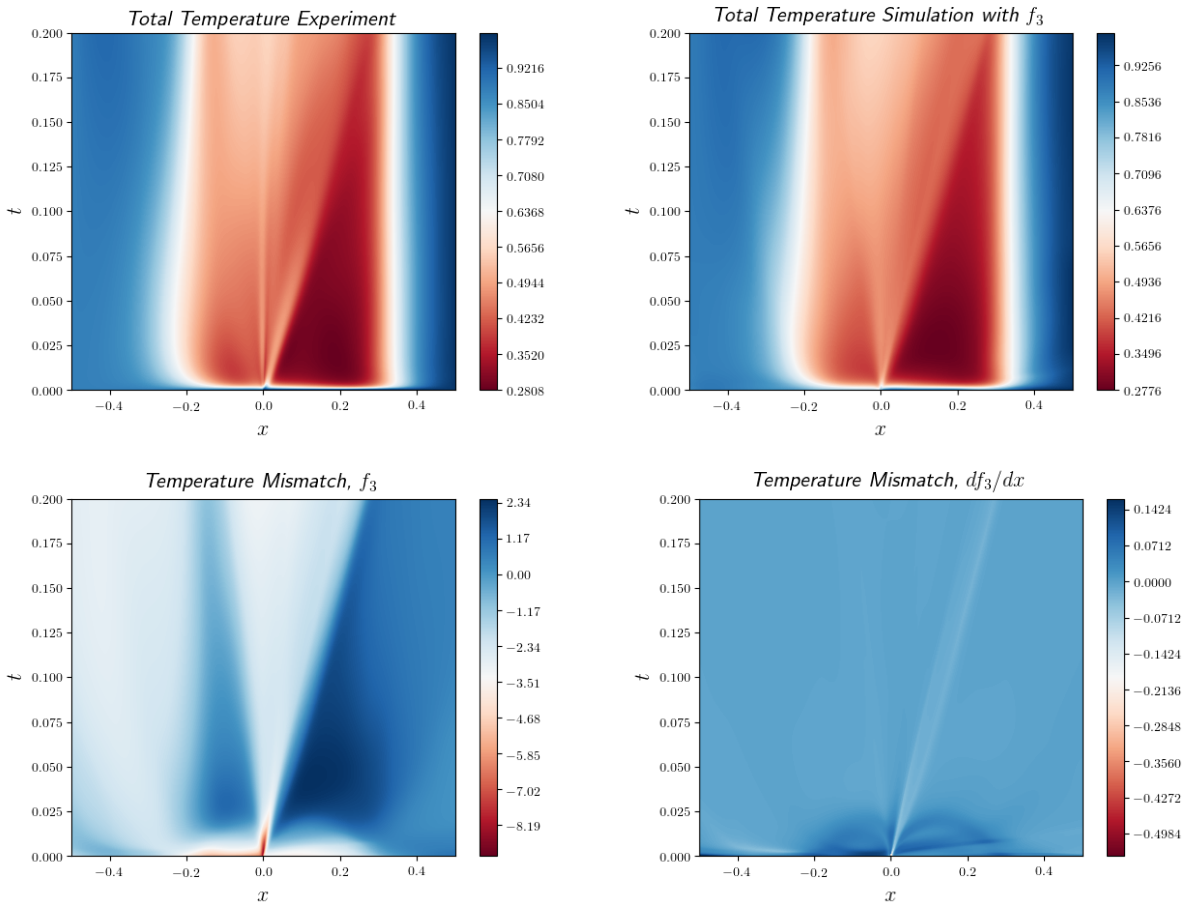

Figure 14: The experimental temperature profile is given on the top left, and the simulation on the top right with f_3. The mismatch function f_3 is shown on the bottom left, with its x-derivative on the bottom right.

where $\rho_{\mathbb{1}}$ is the first chemical species with density ρ_1, and $\rho_{\mathbb{2}}$ the second chemical species with density ρ_2. Here the A_i are excess bulk species, and the k_i are dimensional rate constants. The condition for instability is that $A_2 > A_1^2 + 1$, thus we set $A_2 = 2$, and $A_1 = 0.9$, where for simplicity we set $k_i = 150$ for all i. Photoactivation only occurs in the center of the reactor, and thus

$$\zeta = \frac{1}{12\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}.$$

The solution to this subsystem is shown in Fig. 11

To reveal the mismatch of the underlying system that is actually observed in the experiment, the DNN is trained to model the following enriched PDE system instead of (16):

$$\begin{aligned} \partial_t \rho + (\rho u)_x &= f_1, \\ \partial_t (\rho u) + \left(\rho u^2 + p - \tau \rho u_x\right)_x &= f_2, \\ \partial_t E + \left(Eu + pu - \tau \rho u u_x - \kappa T_x\right)_x &= f_3 \end{aligned} \tag{19}$$

where

$$E = \frac{p}{\gamma - 1} + \frac{\rho}{2} u^2,$$

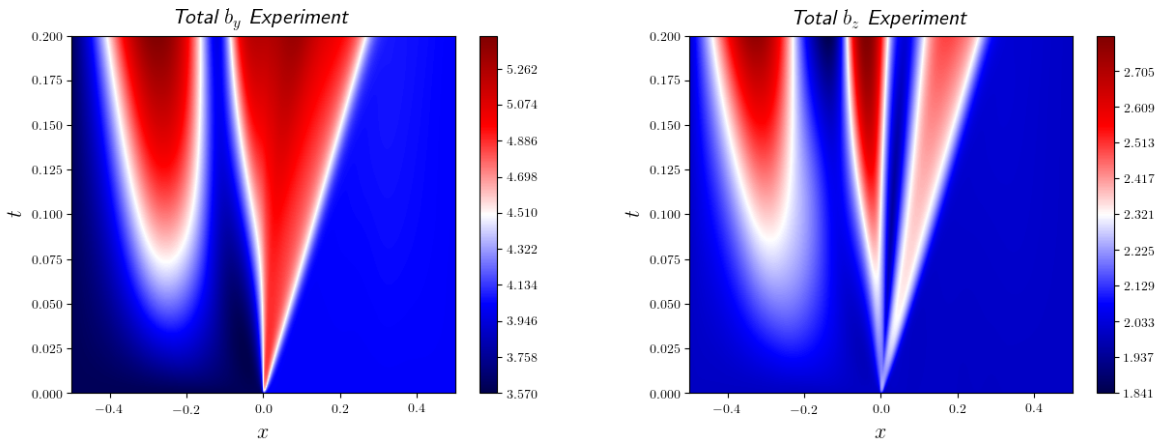

Figure 15: The experimental magnetic fields, where b_x is measured to be negligibly small.

with again the same initial-boundary data from Table 3.

The solution from the experiment (17) is now used as training data to supervise (19). Formally the objective function from (4) is enriched with the training data, $\boldsymbol{U}_{\text{exp}} = (\rho_{\text{exp}}, \rho_{\text{exp}}, u_{\text{exp}}, E_{\text{exp}})$ from (17), by appending the new loss functions

$$\|\rho_{\text{exp}} - \rho\|_{\mathfrak{L}_s(\Omega\times[0,T_s],\mathscr{E})} + \|\rho_{\text{exp}} u_{\text{exp}} - \rho u\|_{\mathfrak{L}_s(\Omega\times[0,T_s],\mathscr{E})} + \|E_{\text{exp}} - E\|_{\mathfrak{L}_2(\Omega\times[0,T_s],\mathscr{E})}$$

to (4), where $\mathscr{E}$ are the experimentally measured data support points, and $\mathfrak{L}_s$ is the mean square error. For simplicity, the support points $\mathscr{E}$ are chosen as a 1000×1000 point grid in (x, t). In addition, the neural network outputs associated to the $f_i = f_i(x, t)$ are L^2-regularized by setting:

$$\sum_i \|f_i^2\|_{\mathfrak{L}_i(\Omega\times[0,T_s],\wp_i)},$$

where here the distributions $\wp_i$ are chosen to be the same as for (19), and the $\mathfrak{L}_i$ are L^2-losses with weight $w_i = 0.0001$ for each i in order to minimize the penalization for accumulating non-zero f_i.

The resulting system matches up reasonably well to the experimental data. As can be seen in Fig. 12, the simulation density and experimental density match to within the eyeball norm, where the initial mismatch is corrected with the f_1 function. Though the total density nearly completely obfuscates the oscillations from the reactive subsystem in Fig. 11, the total density is off primarily near the initial state, where the reactive subsystem indicate a dramatic influx of mass due to the species being tracked by the sensor data in the experiment. This type of mismatch in the mass conservation clearly indicates chemical reactions relative to the tracked species densities, unless, of course, the system is somehow not closed. In Fig. 13 the velocity mismatch is similarly convincing, where the mismatch f_2 shows complicated behavior with nontrivial variation that cannot be readily explained by the mass influx in f_1. Similarly the mismatch f_3 in the temperature in Fig. 14 also shows an unusual signature not present in f_1.

Returning to our list of potential questions about how the model system relates to the experimental system (17), we can now make some fairly strong conclusions. First, clearly there is a mass mismatch in the measured species density. The mismatch is substantial and cannot be captured by simple parameter estimation. Therefore, there is physics in the mass equation that is not being accounted for by the ascribed model system (16). Moreover, it is also straightforward to conclude that the giant

influx of mass makes a chemical reaction a very likely candidate to explain the experimental behavior, assuming the system is closed.

In contrast to the mass equation, the momentum and energy equations in (16) show perturbations away from the model Sod shock solution that indicate that complicated system dynamics is occurring in the continuum. While this behavior might be more challenging to diagnose, it is clear in these cases too that the base equations (16), even with the addition of f_1, cannot account for the system response, and there is missing physics. As it so happens, knowing simply the magnetic field variation of the experiment, as shown in Fig. 15 is really enough to diagnose the mismatch in both the the velocity f_2 and temperature f_3 as related to the magnetic field, since they exhibit signature features that track the magnetic field variation.

5. Conclusion

Gridless representations provided by deep neural networks in combination with numerical optimization can be used to solve complicated systems of differential equations that display irregular solutions. This requires fairly straightforward techniques, and in irregular solutions benefits from the usual trick of adding numerical diffusion to smooth numerically unstable function representations. The DNN method compares favorably with regard to accuracy and stability to more conventional numerical methods and yet enables one-shot exploration of the entire parameter space. The incorporation of efficient optimization algorithms in DNN methods into the PDE solver lends itself to an ease and simplicity of integrating advanced data analytic techniques into physics-enriched model systems, and provides an opportunity for transitioning into a data-rational scientific paradigm. Early results indicate that DNNs enable a simple and powerful framework for exploring and advancing predictive capabilities in science and engineering.

6. Acknowledgements

This work was supported by U.S. DOE Contract No. DE-FG02-04ER54742 and U.S. DOE Office of Fusion Energy Sciences Scientific Discovery through Advanced Computing (SciDAC) program under Award Number DE-SC0018429. This work was also supported by the NSF grant AST-1413501. We acknowledge the Texas Advanced Computing Center at The University of Texas at Austin for providing HPC resources. Computations were performed on the Maverick2 GPU cluster.

[1] Abgrall, R., Saurel, R., APR 10 2003. Discrete equations for physical and numerical compressible multiphase mixtures. Journal of Computational Physics 186 (2), 361–396. 1

[2] Aster, R., Borchers, B., Thurber, C., 2005. Parameter Estimation and Inverse Problems. In: Parameter Estimation and Inverse Problems. Vol. 90 of International Geophysics Series. Academic Press Inc Elsevier Science, pp. 1–303. 1

[3] Babuška, I., Banerjee, U., Osborn, J. E., 2003. Survey of meshless and generalized finite element methods: a unified approach. Acta Numerica 12, 1–125.
URL https://doi-org.ezproxy.lib.utexas.edu/10.1017/S0962492902000090 1

[4] Belytschko, T., Lu, Y. Y., Gu, L., 1994. Element-free Galerkin methods. International Journal for Numerical Methods in Engineering 37 (2), 229–256.
URL https://doi-org.ezproxy.lib.utexas.edu/10.1002/nme.1620370205 1

[5] Berg, J., NystrÃűm, K., 2019. Data-driven discovery of pdes in complex datasets. Journal of Computational Physics 384, 239 – 252.
URL http://www.sciencedirect.com/science/article/pii/S0021999119300944 1, 4.4

[6] Berger, M., Colella, P., May 1989. Local Adaptive Mesh Refinement for Shock Hydrodynamics. Journal of Computational Physics 82 (1), 64–84. 3.1.1

[7] Bochev, P. B., Gunzburger, M. D., 2009. Least-squares finite element methods. Vol. 166 of Applied Mathematical Sciences. Springer, New York.
URL https://doi-org.ezproxy.lib.utexas.edu/10.1007/b13382 1, 2.3.1

[8] Bremer, M., Kazhyken, K., Kaiser, H., Michoski, C., Dawson, C., Aug 2019. Performance comparison of hpx versus traditional parallelization strategies for the discontinuous galerkin method. Journal of Scientific Computing 80 (2), 878–902.
URL https://doi.org/10.1007/s10915-019-00960-z 4.2

[9] Champion, K., Lusch, B., Kutz, J. N., Brunton, S. L., 2019. Data-driven discovery of coordinates and governing equations. 4.4

[10] Chevalier, R., Blondin, J., Emmering, R., Jun 10 1992. Hydrodynamic instabilities in supernova-remnants - self-similar driven waves. Astrophysical Journal 392 (1, 1), 118–130. 1

[11] DeVore, R. A., Lorentz, G. G., 1993. Constructive approximation. Vol. 303 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer-Verlag, Berlin.
URL https://doi.org/10.1007/978-3-662-02888-9 1

[12] Domb, C., Sykes, M. F., Randall, J. T., 1957. On the susceptibility of a ferromagnetic above the curie point. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences 240 (1221), 214–228.
URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1957.0078 2.3.1

[13] Dumbser, M., Balsara, D., Tavelli, M., Fambri, F., 2019. A divergence-free semi-implicit finite volume scheme for ideal, viscous, and resistive magnetohydrodynamics. International Journal for Numerical Methods in Fluids 89 (1-2), 16–42.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4681 4.4.1

[14] Dumbser, M., Kaeser, M., Titarev, V. A., Toro, E. F., Sep 10 2007. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. Journal of Computational Physics 226 (1), 204–243. 3.1

[15] Evans, L. C., 2010. Partial differential equations. American Mathematical Society, Providence, R.I. 2.1

[16] Forrester, A., SoÌĄbester, A., Keane, A., service), W. I. O., 2008. Engineering design via surrogate modelling : a practical guide. Hoboken, N.J. : Wiley ; Chichester : John Wiley [distributor], description based upon print version of record.
URL http://www.SLQ.eblib.com.au/patron/FullRecord.aspx?p=366798 4.2

[17] Fournier, D. A., Skaug, H. J., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M. N., Nielsen, A., Sibert, J., 2012. Ad model builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optimization Methods & Software 27 (2), 233 – 249.
URL http://ezproxy.lib.utexas.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=syh&AN=73326763&site=ehost-live 2.3

[18] Garnett, J. B., 1981. Bounded analytic functions. Vol. 96 of Pure and Applied Mathematics. Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York-London. 1

[19] Gerstner, T., Griebel, M., 1998. Numerical integration using sparse grids. Numerical Algorithms 18 (3-4), 209–232.
URL https://doi.org/10.1023/A:1019129717644 1

[20] Ghanem, R., Higdon, D., Owhadi, H., 2017. Handbook of uncertainty quantification. Springer, New York. 4.4

[21] Giannakouros, j., Karniadakis, G., MAR 30 1992. Spectral Element Fct Method for Scalar Hyperbolic Conservation-laws. International Journal for Numerical Methods in Fluids 14 (6), 707–727. 3.1

[22] Glorot, X., Bengio, Y., 13–15 May 2010. Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y. W., Titterington, M. (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Vol. 9 of Proceedings of Machine Learning Research. PMLR, Chia Laguna Resort, Sardinia, Italy, pp. 249–256.
URL http://proceedings.mlr.press/v9/glorot10a.html 2.3

[23] Gordon, N., Salmond, D., Smith, A., APR 1993. Novel-approach to nonlinear non-gaussian Bayesian state estimation. IEE Proceedings-F Radar and Signal Processing 140 (2), 107–113. 1

[24] Hadamard, J., 1902. Sur les problèmes aux dérivés partielles et leur signification physique. Princeton University Bulletin 13, 49–52. 2.1

[25] Han, J., Jentzen, A., E, W., 2018. Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences 115 (34), 8505–8510.
URL https://www.pnas.org/content/115/34/8505 1

[26] Hesthaven, J. S., Warburton, T., 2008. Nodal discontinuous Galerkin methods. Vol. 54 of Texts in Applied Mathematics. Springer, New York, algorithms, analysis, and applications.
URL https://doi.org/10.1007/978-0-387-72067-8 1

[27] Huber, P. J., 03 1964. Robust estimation of a location parameter. The Annals of Mathematical Statistics 35 (1), 73–101.
URL https://doi.org/10.1214/aoms/1177703732 2.3

[28] Isett, P., NOV 2018. A proof of Onsager's conjecture. Annals of Mathematics 188 (3), 871–963. 2.1

[29] Jiang, B.-N., Carey, G. F., 1990. Least-squares finite element methods for compressible Euler equations. International Journal for Numerical Methods in Fluids 10 (5), 557–568.
URL https://doi-org.ezproxy.lib.utexas.edu/10.1002/fld.1650100504 1

[30] Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. 2.3

[31] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., 1983. Optimization by simulated annealing. Science 220 4598, 671–80. 4.1

[32] Lagaris, I. E., Likas, A., Fotiadis, D. I., 1998. Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks 9 (5), 987–1000. 1

[33] Lax, P. D., 1973. Hyperbolic systems of conservation laws and the mathematical theory of shock waves. Society for Industrial and Applied Mathematics, Philadelphia, Pa., conference Board of the Mathematical Sciences Regional Conference Series in Applied Mathematics, No. 11. 1

[34] LeVeque, R., MAR 1 1997. Wave propagation algorithms for multidimensional hyperbolic systems. Journal of Computational Physics 131 (2), 327–353. 3.1.1

[35] LeVeque, R. J., 2002. Finite-Volume Methods for Hyperbolic Problems. Cambridge University Press. 2.3.2

[36] Li, X., 2016. Error estimates for the moving least-square approximation and the element-free Galerkin method in n-dimensional spaces. Applied Numerical Mathematics. An IMACS Journal 99, 77–97.
URL https://doi-org.ezproxy.lib.utexas.edu/10.1016/j.apnum.2015.07.006 1

[37] Long, Z., Lu, Y., Ma, X., Dong, B., 2018. PDE-net: Learning PDEs from data.
URL https://openreview.net/forum?id=SylJ1D1C- 4.4

[38] Lu, Z., Pu, H., Wang, F., Hu, Z., Wang, L., 2017. The expressive power of neural networks: A view from the width. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 6231–6239.
URL http://papers.nips.cc/paper/7203-the-expressive-power-of-neural-networks-a-view-fro
pdf 2.3.1

[39] Mabuza, S., Shadid, J. N., Kuzmin, D., MAY 15 2018. Local bounds preserving stabilization for continuous Galerkin discretization of hyperbolic systems. Journal of Computational Physics 361, 82–110. 3.1

[40] Malaya, N., McDougall, D., Michoski, C., Lee, M., Simmons, C. S., 2017. Experiences porting scientific applications to the intel (knl) xeon phi platform. In: Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact. PEARC17. ACM, New York, NY, USA, pp. 40:1–40:8.
URL http://doi.acm.org/10.1145/3093338.3093371 4.2

[41] Mercer, G., Roberts, A., 1990. A Centre Manifold Description of Contaminant Dispersion in Channels with Varying Flow Properties. SIAM Journal on Applied Mathematics 50 (6), 1547–1565.
URL https://doi.org/10.1137/0150091 2.3.1

[42] Michoski, C., Chan, J., Engvall, L., Evans, J., 2016. Foundations of the blended isogeometric discontinuous galerkin (bidg) method. Computer Methods in Applied Mechanics and Engineering 305, 658 – 681.
URL http://www.sciencedirect.com/science/article/pii/S0045782516300457 3.1.1

[43] Michoski, C., Dawson, C., Kubatko, E. J., Wirasaet, D., Brus, S., Westerink, J. J., Jan 2016. A Comparison of Artificial Viscosity, Limiters, and Filters, for High Order Discontinuous Galerkin Solutions in Nonlinear Settings. Journal of Scientific Computing 66 (1), 406–434. 3.1

[44] Michoski, C., Mirabito, C., Dawson, C., Wirasaet, D., Kubatko, E. J., Westerink, J. J., Sep 10 2011. Adaptive hierarchic transformations for dynamically p-enriched slope-limiting over discontinuous Galerkin systems of generalized equations. Journal of Computational Physics 230 (22), 8028–8056. 3.1, 3.3.1

[45] Nabian, M. A., Meidani, H., 09 2019. Physics-Driven Regularization of Deep Neural Networks for Enhanced Engineering Design and Analysis. Journal of Computing and Information Science in Engineering 20 (1), 011006.
URL https://doi.org/10.1115/1.4044507 1

[46] Ohwada, T., Kobayashi, S., JUN 10 2004. Management of discontinuous reconstruction in kinetic schemes. Journal of Computational Physics 197 (1), 116–138. 1

[47] Oliver, T. A., Terejanu, G., Simmons, C. S., Moser, R. D., 2015. Validating predictions of unobserved quantities. Computer Methods in Applied Mechanics and Engineering 283, 1310 – 1335.
URL http://www.sciencedirect.com/science/article/pii/S004578251400293X 4.4

[48] Peherstorfer, B., Willcox, K., Gunzburger, M., 2018. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. SIAM Review 60 (3), 550–591. 4.2

[49] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., Courville, A., 2018. On the spectral bias of neural networks. 3.1

[50] Raissi, M., Jan. 2018. Deep hidden physics models: Deep learning of nonlinear partial differential equations. J. Mach. Learn. Res. 19 (1), 932–955.
URL http://dl.acm.org/citation.cfm?id=3291125.3291150 4.4

[51] Raissi, M., Perdikaris, P., Karniadakis, G., 2019. 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.
URL http://www.sciencedirect.com/science/article/pii/S0021999118307125 1, 2.3, 4.4

[52] Rudy, S. H., Brunton, S. L., Proctor, J. L., Kutz, J. N., 2017. Data-driven discovery of partial differential equations. Science Advances 3 (4).
URL http://advances.sciencemag.org/content/3/4/e1602614 4.4

[53] Schaeffer, H., 2017. Learning partial differential equations via data discovery and sparse optimization. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 473 (2197), 20160446.
URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2016.0446 4.4

[54] Sirignano, J., Spiliopoulos, K., 2018. DGM: A deep learning algorithm for solving partial differential equations. Journal of Computational Physics 375, 1339 – 1364.
URL http://www.sciencedirect.com/science/article/pii/S0021999118305527 1, 2.3, 4.3, 4.3

[55] Smoller, J., 1983. Shock waves and reaction-diffusion equations. Vol. 258 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Science]. Springer-Verlag, New York-Berlin. 1

[56] Sod, G. A., 1978. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. Journal of Computational Physics 27 (1), 1 – 31.
URL http://www.sciencedirect.com/science/article/pii/0021999178900232 1

[57] Strang, G., Fix, G. J., 1973. An analysis of the finite element method. Prentice-Hall, Inc., Englewood Cliffs, N. J., prentice-Hall Series in Automatic Computation. 1

[58] Wang, L., Yao, F., Zhou, S., Jia, H., 2009. Optimal regularity for the poisson equation. Proceedings of the American Mathematical Society 137 (6), 2037–2047.
URL http://www.jstor.org/stable/20535956 2.1

[59] Xu, Z., Sep 2014. Parametrized Maximum Principle Preserving Flux Limiters for High Order Schemes Solving Hyperbolic Conservation Laws: One-dimensional Scalar Problem. Mathematics of Computation 83 (289), 2213–2238. 3.1

[60] Xu, Z., Liu, Y., Shu, C.-W., Sep 1 2009. Hierarchical reconstruction for spectral volume method on unstructured grids. Journal of Computational Physics 228 (16), 5787–5802. 3.1

[61] Zhang, Z., Gogos, G., Jun 2004. Theory of shock wave propagation during laser ablation. Physical Review B 69 (23). 1