ABSTRACT

Physics-informed neural networks (PINNs) seamlessly integrate data and physical constraints into the solving of problems governed by differential equations. In settings with little labeled training data, their optimization relies on the complexity of the embedded physics loss function. Two fundamental questions arise in any discussion of frequently reported convergence issues in PINNs: Why does the optimization often converge to solutions that lack physical behavior? And why do reduced domain methods improve convergence behavior in PINNs? We answer these questions by studying the physics loss function in the vicinity of fixed points of dynamical systems. Experiments on a simple dynamical system demonstrate that physics loss residuals are trivially minimized in the vicinity of fixed points. As a result we observe that solutions corresponding to nonphysical system dynamics can be dominant in the physics loss landscape and optimization. We find that reducing the computational domain lowers the optimization complexity and chance of getting trapped with nonphysical solutions.

Keywords  Physics-Informed Neural Networks, Dynamical Systems, Fixed Points, Convergence Issues

1 Introduction

Dynamical systems are governed by differential equations and are ubiquitous in many scientific disciplines such as economics, biology, physics and engineering. The upsurge in scientific deep learning has led to the development of neural network approaches that are applicable to those systems and often superior to classical/numerical methods. One way to study dynamical systems is by employing data-based or supervised approaches. State-of-the-art methods infer (at least) some part of the physics from labeled data and the underlying physical process that generates it. In contrast, physics-informed neural networks (PINNs) directly incorporate prior physical knowledge by minimizing residuals of the underlying differential equation. PINNs are the prime paradigm of physics-informed machine learning and are capable of being trained with only a few labeled data points. Their mesh-free, time-continuous and unsupervised encoding of differential equations renders the integration of data and physical constraints seamless.

∗corresponding author

1 code is available at https://github.com/frohrhofer/PINN_TF2
Figure 1: **Nonphysical System Dynamics in PINNs.** The underlying dynamical system in our study is given by $f(y) = y \cdot (1 - y^2)$ and exhibits three fixed points, which to close trajectories are either attractive (green circles) or repulsive (red circle). The physics loss optimization minimizes residuals of the governing differential equation and often suffers from convergence issues that lead to incorrectly learned system dynamics. In this work, we tackle the root cause of those commonly observed convergence issues and nonphysical system dynamics in PINNs.

Those assets have pushed PINNs into a vast number of applications for dynamical systems [7, 8], including system identification [9, 10], hidden state inference [11, 12] and surrogate modeling [13, 14].

In forward problem-solving, labeled data is only needed to encode initial and boundary conditions. For a complete and unique problem definition, data and physical constraints are considered either soft and in a multi-objective manner or hard through the use of prior dictionaries [15]. Both variants, however, rely on the optimization of the underlying physics loss function using gradient descent variants. Optimization success and accuracy typically depend on the complexity of the studied system and the corresponding physics loss function. Convergence issues are common in training deep neural networks [16, 17], and PINNs seem to suffer from them regularly.

### 1.1 Related Work

**Convergence Issues.** Issues in optimization of PINNs often relate to incorrectly learned system dynamics (see Figure 1). Conflicting objectives in multi-objective optimization have been identified as a common root cause of convergence issues [18]. The importance of objectives in view of the Pareto front has been analyzed in [19]. Part of the aforementioned study are the effects of system parameters such as the computational domain, and efficiency of adaptive loss weighting schemes [20, 21]. In general, adaptive approaches have become popular in the optimization of PINNs [22, 23], including adaptive activation functions [24, 25] and soft attention mechanisms [26, 27]. Those approaches usually improve the convergence speed and/or reliability of PINNs. The physics loss function and optimization have been directly studied several times [28, 29] and from both a theoretical [30, 31] and system dependent perspective [32]. In [33] further failure modes of PINNs are shown and discussed as a consequence of the PINN’s setup which makes the loss landscape very hard to optimize. Our work focuses on bridging the gap between several of these aspects and providing new insights into convergence issues in PINNs.

**Reduced Domain Methods.** When it comes to solving differential equations on complex spatio-temporal domains, reduced domain approaches have gained popularity in PINNs. In [34] the authors follow several time-adaptive approaches; in one implementation, multiple combined and coupled PINN instances are considered, each of which is applied on a subdomain. Consequently, the entire (time-)domain is covered by individual instances which interact via the respective neighboring segments. Likewise, [33] discusses a method called "sequence-to-sequence learning", whose performance is superior to learning the entire spatio-temporal domain. A similar approach is provided in [35], which introduces so-called "X-PINNs" or "extended physics-informed neural networks". Those PINNs, among further implementations [36, 37], can be applied to arbitrary complex-geometry domains in multiscale and multiphysics problems [38]. In our study, we borrow the common idea of reducing the computational domain and explain why these methods work.
1.2 Our Contribution

In this work, we experimentally investigate the physics loss function and study its optimization complexity based on a simple dynamical system (see Figure 1 and Section 3). A brief introduction to PINNs and theory of dynamical systems is found in Section 2. In the course of our study, we identify under which conditions convergence issues in the PINN optimization may appear (Section 4.1 and 4.4). We relate incorrectly learned system dynamics to suboptimal solutions which appear in the physics loss landscape and correspond to fixed points of the underlying system (Section 4.2). We find that in the vicinity of those fixed points physics loss residuals are trivially minimized which renders certain nonphysical solutions economical in the physics loss optimization. Furthermore, we visually capture the change in optimization complexity when the computational domain is reduced (Section 4.3). Finally, in Section 5 we provide a concluding picture for commonly observed convergence issues in PINNs and the success of reduced domain approaches.

2 Background

In this work we are concerned with simulating the dynamics of physical systems that are governed by ordinary differential equations of the form

\[ \frac{dy}{dt} = f(t, y(t)), \quad t \in [0, T], \]

where \( y \) denotes the unknown solution function and \( f \) is a given (nonlinear) function with \( f : \Omega \subset \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n \). Here, the (only) independent variable is expressed by \( t \), which often relates to time in dynamical systems. Furthermore, \( t \in [0, T] \) specifies the computational domain where \( T \) can be interpreted as simulation time, as it will be referred to in the course of this work (unless explicitly stated otherwise). Further details on dynamical systems governed by autonomous differential equations are provided in Section 2.2.

2.1 Physics-Informed Neural Networks

Fully-connected neural networks (FC-NNs) have been the primary workhorse of PINNs due to their fair trade-off between simplicity and expressive power \(^1\). Likewise, we consider a FC-NN to approximate the unknown solution function of Equation 1, i.e., \( y(t) \approx y(t; \theta) \equiv y_\theta(t) \) with \( \theta \in \mathbb{R}^{n_\theta} \). Smooth activation functions are preferred; common choices are the hyperbolic tangent (tanh) or Sigmoid linear unit (SiLU, swish), which render the approximated solution function smooth \(^2\).

A key element of physics-informed machine learning is automatic differentiation (AD) \(^3\), the algorithm that typically retrieves gradients of the loss function with respect to network parameters. Here, the same algorithm is used to obtain the derivative of the network’s output with respect to its input, i.e., \( dy/dt \). In order to retrieve the derivatives, AD demands particular collocation coordinates to be passed through the network in a feed-forward operation. Those coordinates are referred to as collocation points and define the data set, here \( \{ t_i \}_{i=1}^{N_{\text{col}}} \), for penalizing residuals of the differential equation. The residuals as well as their minimization through the physics loss function are defined according to

\[ L(\theta; T) = \frac{1}{N_{\text{col}}} \sum_{i=1}^{N_{\text{col}}} \left( \frac{dy_\theta}{dt} \bigg|_{t_i} - f(t_i, y_\theta(t_i)) \right)^2, \]

with \( \{ t_i \}_{i=1}^{N_{\text{col}}} \) sampled from the computational domain \( t \in [0, T] \). Since collocation points do not need any label attached to them, the data set can be sampled in a fully unsupervised manner prior network training or anew at each training epoch. Furthermore, we explicitly consider the dependency on the computational domain, here \( L(\cdot; T) \), which is essential to our experiments.

2.2 Autonomous Differential Equations and Stability

As a subclass of ordinary differential equations, autonomous differential equations do not explicitly depend on time, i.e., the (nonlinear) function \( f \) in Equation 1 is given by \( f \equiv f(y(t)) \). Examples are the Navier-Stokes equations describing fluid flow or systems found in Hamiltonian mechanics. Without the loss of generality, autonomous differential equations exhibit certain fixed points or equilibrium points similar to what can be found in any dynamical system. Those fixed

---

\(^1\)Thus mesh-free and time-continuous in the context of differential equations.

\(^2\)As part of modern deep learning libraries, AD can also retrieve the full Jacobian matrix, which is used to implement partial differential equations.
points are characterized as being either attractive or repulsive to nearby trajectories. According to their definition, fixed points appear as zeros of the function \( f \), i.e.,
\[
\frac{dy}{dt} = f(y^\pm) = 0 ,
\]
holds true on \( \Omega \) for any fixed point \( y^\pm \). Attractive fixed points, here \( y^+ \), are considered as (asymptotically) stable in the sense of Lyapunov stability \[40\]. Any trajectory \( y(t) \) close to the attractive fixed point will either stay close to it (stable) or further converge to it as \( t \to \infty \) (asymptotically stable). Repulsive fixed points \( y^- \), in contrast, are unstable and any close trajectory will move away from it as \( t \to \infty \). A typical analogy for an unstable fixed point is a perfectly balanced, upright pendulum that diverges from its unstable position in reaction to the smallest perturbation. A pendulum that perfectly hangs downwards is a stable fixed point that is asymptotically stable when air resistance is taken into account.

3 Experimental Setting

3.1 System Settings

Our study focuses on a simple autonomous differential equation given by
\[
\frac{dy}{dt} = f(y(t)) = y \cdot (1 - y^2) ,
\]
with \( t \in [0, T] \) and a total of three zeros or fixed points at \( y^\pm \in \{-1, 0, 1\} \). Asymptotically stable fixed points are located at \( y^+ = \pm 1 \), whereas an unstable fixed point is located at \( y^- = 0 \) (see Figure 1).

Initial conditions are critical to defining a well-posed problem, and for the aforementioned system, they are given by
\[
y(t_0) = y_0 ,
\]
with \( (t_0, y_0) \in \Omega \), a point in the domain of \( f \). The data tuple \( (t_0, y_0) \), here \( t_0 = 0 \), determines the unique evolution of a trajectory \( y(t) \) and can be encoded either implicitly through soft constraints (mean squared error optimization) or explicitly through prior dictionaries \[15\]. Better known as hard constraints \[41\], the latter promote specially designed network architectures that are in accordance with the initial conditions by definition.

For hard constraining of the network function \( y_0 \), we use the following approach
\[
\hat{y}(t) = y_0 + d(t) \cdot y_0(t) ,
\]
with \( \hat{y} \) denoting the output of the new network and \( d \) the so-called distance function. In general, the distance function can be chosen arbitrarily but has to fulfill \( d(t_0) = 0 \) and \( d(t > t_0) \neq 0 \), so that Equation 6 automatically meets the initial condition \[3\]. Commonly selected distance functions are linear, i.e., \( d(t) = t - t_0 \), or the hyperbolic tangent \( d(t) = \tanh(t - t_0) \).

3.2 Network Settings

All experiments are conducted using a standard 2x20 FC-NN architecture with a Sigmoid linear unit (swish-1) activation function. The output layer uses a linear activation function and the hyperbolic tangent is selected as a distance function (see Section 3.1). The network weights are initialized using Glorot uniform initialization \[17\]. The variant of stochastic gradient descent (SGD) optimization that we use is Adam \[42\] with default settings for the learning rate \( (\alpha = 0.001) \) and exponential decay rates \( (\beta_1 = 0.9, \beta_2 = 0.999) \) as implemented in TensorFlow 2 \[43\]. The training duration for any model instance is set to 50000 epochs. Collocation points are uniformly sampled anew at each training epoch with a batch size of \( N_{col} = 1024 \) data points. Further details on the setup are defined in the individual experimental sections \[3.1\] through \[4.4\].

In preliminary experiments, we assessed the outcome of different design choices. We were able to qualitatively reproduce all of the results presented in the next section independent of the selected experimental setup. For demonstration purposes, experiments presented in the main part of this work are performed using the settings stated above.

4 Understanding the Difficulty of Training Physics-Informed Neural Networks

In this section we present a collection of experiments which reflect the identification, analysis and understanding of difficulties in training PINNs on dynamical systems. Code for the basic model is available at https://github.com/frohrhofer/PINN_TP2.
4.1 Identification of Convergence Issues

In this first experiment, we assess and compare the SGD convergence in our experimental setup by means of a (i) physics-driven and (ii) data-guided approach. The former (i) does not use any labeled data. The optimization is purely unsupervised and conforms to the optimization of the physics loss function given by Equation 2. In this approach, the SGD convergence is completely determined by the physics loss optimization and its complexity. In the latter (ii), the optimization is supported by as few as ten labeled training examples that are equally spaced along the computational domain and taken from the analytical solution. Here, the optimization conforms to an unweighted multi-objective optimization of the physics loss and the mean squared error loss, embedding the labeled training examples. The multi-objective optimization stops after 25000 epochs and continues with the physics loss optimization only for a further 25000 epochs. With this setting, the initially seen training examples guide the SGD optimization into the basin of attraction of the analytical solution. This ensures that the subsequent physics loss optimization converges to minima that correspond to the analytical solution.

In particular, we train multiple PINN instances with different initial values and seeds for the weight initialization. The initial values $y_0$ are chosen on a fixed logarithmic scale with increasing distance to the unstable solution in both directions so that $|y_0 - y^-| \in \{10^{-4}, \ldots, 10^0\}$ where $y^- = 0$. For each initial value and both approaches (i) and (ii), five unique seeds are used for the weight initialization. The simulation time is set to $T = 10$. Predicted trajectories and corresponding learning curves of the physics loss optimization are found in Figure 2.

From Figure 2(a), it is apparent that only a few of the physics-driven PINN instances converge to the analytical solution, which can certainly be judged by trajectories approaching the unstable solution at $y^- = 0$. This behavior is nonphysical and would not be observed in nature or using classical methods such as Runge-Kutta. The attraction to the unstable solution becomes more consistent the closer trajectories are initialized to it, i.e., as $|y_0 - y^-| \to 0$. Furthermore, the attraction is even apparent directly at initializations at one of the two stable solutions, i.e., when $y_0 = y^+ \pm 1$. The corresponding learning curves indicate that the SGD optimization potentially becomes trapped at poor locations in the loss landscape characterized by high physics losses. In Figure 2(b), on the other hand, the data-guided PINN instances perform well. This is indicated by the trajectories converging to the respective stable solutions and coinciding with the analytical solution (within a certain accuracy, not shown). In addition, the corresponding learning curves reveal a healthy SGD optimization with no serious effects at the transition to optimization (dashed line). This indicates that the SGD optimization has indeed converged to minima that correspond to the analytical solution. If we now compare Figure 2(a) and 2(b), we observe that physics loss values of the physics-driven approach can be lower in value than those of the data-guided approach.

Thus in Figure 2(c), we plot the lowest physics loss values $L_{\text{min}}$ across all epochs for each PINN instance. We plot those values as a function of initialization distance to the unstable solution in order to experimentally assess the depth of
dominant locations in the optimization landscape. From the figure, it is apparent that the optimization and convergence of the data-guided PINN instances (blue pluses) are consistent with a slight tendency to higher loss values as trajectories are initialized closer to the unstable solution. In contrast, the optimization of the physics-driven instances (orange crosses) is rife with irregularities. Trajectories initialized at one of the stable solutions, i.e., at $|y_0 - y^-| = 1$, partly converge to the analytical solution with loss values similar to those of the data-guided instances. However, some physics-driven instances – again, the only difference is the initialization of network weights – suffer from convergence issues indicated by comparable poor minimal loss values. We identify those scenarios as simulated trajectories that approach the unstable solution in Figure 2(a). The observed convergence issues become more prominent as trajectories are initialized closer to the unstable solution, i.e., as $|y_0 - y^-| \to 0$. Here, we note that the distinct loss levels of the physics-driven instances as well as those of the plateaus in the corresponding learning curves arise from the network learning the trivial zero solution function, which approximately leaves behind the squared residuals of the initial values, i.e., $L_{\text{min}} \approx y_0^2$. This trivial error propagation is of particular interest (and will be discussed later) as it becomes so economical that solutions corresponding to nonphysical system dynamics form the global minima of the optimization.

In general, these results demonstrate that incorrectly learned system dynamics do not necessarily go hand in hand with large observed physics loss values. What is more, the results demonstrate that under certain circumstances the optimization hardly yields accurate results, as the global minimum of the physics loss landscape potentially does not conform to the analytical solution of the underlying problem.

### 4.2 Suboptimal Solutions in the Physics Loss Landscape

In the previous section, we identified convergence issues as distinct plateaus in the learning curves that are innately related to incorrectly learned system dynamics. During our experiments, we observed a potential escape from those undesired cases which were apparent as distinct drops in the recorded learning curves. One explanation for those observations is provided by the random weight initializations and newly sampled collocation points at each training epoch: Due to the stochastics thus induced, gradients out of the suboptimal locations may appear at a certain point of the training and lead to an escape. Moment settings in modern optimizers may affect the point of escape, but this was not tested in the course of this work. To further investigate these observations, we study the SGD optimization of a single PINN instance that barely managed to escape from a suboptimal location. Here and in the remainder of this work, we take a physics-driven approach only: No labeled training data is used in any of the following experiments.

In this experiment, we train a single PINN instance with an initial value of $y_0 = 0.5$ and simulation time of $T = 8$. Training and optimizer settings are adopted from the previous experiments. Model weights are recorded at discrete time steps, here with an increment of 5000 epochs. In this and the following section, the conducted visualization of the loss
As discussed in the beginning of this work, our major focus is the effects of reduced computational domain approaches. Again, those observations make clear and highlight how solutions corresponding to nonphysical system dynamics can become less economical due to the relative weight with the initial condition.

Several characteristics can be discovered in the collection of physics loss landscapes. On the one hand, we observe that the suboptimal locations corresponding to nonphysical system dynamics (see Figure 3) gradually disappear as the computational domain is reduced from $T = 8$ to $T = 2$. At the same time, the minimum of the analytical solution does not change much in position and depth. In general, we discover a reduced complexity of the physics loss landscape at lower $T$ values, which is indicated by the landscape turning smooth without the appearance of disruptive suboptimal locations. We believe that this can be explained by the fact that the initial physics violation (as apparent in Figure 3) becomes less economical due to the relative weight with the initial condition.

### 4.3 Reduced Computational Domain Effects

As discussed in the beginning of this work, our major focus is the effects of reduced computational domain approaches. In our example, the computational domain is given by $[0, T]$ and can be adjusted by the simulation time $T$. In this section, we visually study the effects on the physics loss landscape and its complexity when the computational domain is limited to a certain simulation time.

For this experiment, we consider a reduced computation domain by sampling collocation points up to $T = \{2, 4, 6, 8\}$ with $N_{col} = 1024$. In particular, we adapt the two weight directions of the previous experiment, thus considering $L_F(\vec{\theta}_1, \vec{\theta}_2; T)$ by evaluating the physics loss function $2$ on the newly sampled collocation data. The outcome of this experiment is found in Figure 4.

In the figure, the loss landscape for $T = 8$ is in accordance with the results already shown in the bottom panel of Figure 3. For clarity, we also highlight the SGD path followed in the previous experiment as a black dashed line. Several characteristics can be discovered in the collection of physics loss landscapes. On the one hand, we observe that the suboptimal locations corresponding to nonphysical system dynamics (see Figure 3) gradually disappear as the computational domain is reduced from $T = 8$ to $T = 2$. At the same time, the minimum of the analytical solution does not change much in position and depth. In general, we discover a reduced complexity of the physics loss landscape at lower $T$ values, which is indicated by the landscape turning smooth without the appearance of disruptive suboptimal locations. We believe that this can be explained by the fact that the initial physics violation (as apparent in Figure 3) becomes less economical due to the relative weight with the initial condition.

Figure 4: **Reduced Computational Domain Effects.** Physics loss landscape evaluated in different computational domains, limited by the simulation time $T$ which decreases from left to right. For clarity, the SGD path followed in the previous experiment is shown as a black dashed line.
Figure 5: Basin of Attraction and Rate of Success. Up to 3000 PINN instances were trained using a random initial value $y_0$, simulation time $T$ and unique seed for weight initialization. (a) Sample of 500 predicted trajectories after completed training; (b) averaged minimal physics loss found by the SGD optimization, subject to simulation time and initial value; (c) rate of (convergence) success. A heuristic method was used to detect whether the SGD optimization has converged to the analytical solution.

In summary, these observations indicate that the physics loss optimization can be simplified and facilitated by reducing the computational domain. In addition, this provides an explanation for why reduced domain approaches are often superior to learning the entire spatio-temporal domain.

4.4 Basin of Attraction and Rate of Success

In our final experiment, we seek to consolidate previous observations by comparing the sizes of basins of attraction in the loss landscape subject to the initial value $y_0$ and simulation time $T$. We train and assess several PINN instances with a random initial value, simulation time and seed for weight initialization. Subsequently, we post-process and show quantities of interest which contribute to a better understanding of which settings cause the optimization to suffer from convergence issues and yield nonphysical system dynamics.

Within the collection of PINN instances, each one is trained using a random initial value $y_0 \in [-1.5, 1.5]$, simulation time $T \in [0.1, 10]$ and unique seed for the weight initialization. We train a total of approximately 3000 instances using the default experimental settings. For post-processing, all recorded runs are collected in predefined bins subject to initial value and simulation time. We ensured that each bin captures at least ten instances. The results of this experiment are found in Figure 5.

Figure 5(a) illustrates predicted trajectories after completed training; for clarity, we have only used a sample of 500 instances. Minimal physics loss values (as evaluated in Section 4.1) were determined and used according to a color code. As anticipated from previous experiments, trajectories optimized and predicted for longer simulation times (here approx. $T > 4$) predominantly tend to approach the unstable fixed point at $y^- = 0$. Shorter trajectories, i.e., optimized and predicted on a reduced computational domain, seem to be learned correctly. This is evident from the trajectories approaching/converging to one of the two stable fixed points $y^+ = \pm 1$ with comparable low physics loss values.

Figure 5(b) shows the corresponding (averaged) minimal physics loss values as a function of the simulation time and initial value. In this plot, we observe a distinct front of transition from low to high loss values, which generally appears at larger simulation times ($T > 5$). The front starts at an initialization close to the unstable fixed point and spreads out as the simulation time increases. At longer simulation times beyond the transition, i.e., in the high loss regime, average losses seem to decrease as trajectories are initialized closer to the unstable fixed point, i.e., as $|y_0 - y^-| \to 0$. This behavior has already been captured and observed in our first experiment and Figure 2. Further insights into this observation are provided by Figure 5(c), which captures the rate of (convergence) success. Here, we used a heuristic method that examines the slope of a trajectory to detect whether it is approaching the unstable fixed point or not, thus quantifying how likely the optimization will succeed with the respective settings. We clearly observe that although initializations close to the unstable fixed point have low physics losses, they have a low rate of success ($\approx 0$) for $T > 5$. Not surprisingly, the rate of success is the highest ($= 1$) for short simulation times and initializations far from the unstable fixed point.
5 Summary, Discussion and Impact

In this section, we briefly summarize what has been observed and seek to give a comprehensive explanation of the fundamental problem underlying the observed convergence issues. We want to address (once more) the advantages of physics-informed machine learning and the contribution this work may provide to further modifications/improvements of PINNs. In general, PINNs allow the consideration of differential equations with a fully mesh-free, time-continuous and unsupervised approach. These advantages eliminate the need for cumbersome generation of a computational mesh as well as the limitations due to discrete-time considerations. Furthermore, data constraints may be seamlessly integrated into multiscale and multiphysics problems at any point in space and time. As we were able to demonstrate in our work, however, PINNs still suffer from convergence issues which lead to nonphysical system dynamics.

Our study and results revealed that fixed points of dynamical systems, irrespective of whether they are attractive or repulsive, exhibit a minimum in the physics loss landscape. For repulsive, or unstable, fixed points, this behavior is undesirable as the SGD optimization potentially is trapped with solutions that approach the unstable fixed point, hence, that lack physical behavior. This was demonstrated by a set of experimental investigations of a simple autonomous differential equation (see Section 3.1). We used the standard physics-informed approach (Section 2.1) to optimize the embedded physics loss function and observed convergence issues while adjusting the initial value and simulation time (Section 4.1 and 4.4). The convergence issues were traced back to suboptimal locations in the optimization landscape that were inextricably related to the unstable fixed point of the studied system (Section 4.2). Furthermore, initial values very close to the fixed point revealed that nonphysical solutions can even conform to the global minimum of the physics loss optimization (Section 4.1). We also observed that suboptimal locations disappear and alleviate the optimization complexity when the computational domain is reduced (Section 4.3).

Fixed points are common in dynamical systems. Some fixed points are easily identified and can be effortlessly studied as shown in this work or found in harmonic oscillator equations. Other fixed points, however, can be latent and difficult to access due to the complexity of the dynamical system. As an example, we refer to the well-known process of vortex shedding in fluid flow which occurs as a result of instabilities close to an unstable fixed point in the Navier-Stokes equations. Regardless of the dynamical system, fixed points appear as zeros in the underlying differential equation (as stated in Equation 8 and illustrated in Figure 1). As part of the physics loss function, residuals of the differential equation are thus small by definition since \( \frac{dy}{dt} = f(y) \approx 0 \) in the vicinity of fixed points. These characteristics seem to contribute to a certain simplicity of minimizing the residuals in the physics loss function. The physics loss function would be trivially fulfilled if all individual components were kept close to zero. Loosely speaking, there is certain attractiveness to "dispose" trajectories to close fixed points where residuals are kept small in the simplest way, thus formalizing Occam’s razor. We were able to capture this in Figure 2 and Figure 3.

The literature recognizes common convergence issues in training PINNs and proposes several remedies. In particular, we have observed that methods as simple as limiting the computational domain have a major impact on the simulated system dynamics and SGD convergence. A reduction in the computational domain showed that suboptimal locations in the loss landscape will disappear and a reduced complexity of the physics loss optimization can be achieved. This observation explains the success of reduced domain methods in PINNs.

Although simple and effective in their feasibility, these methods have their own downsides: In practical realizations (see Section 1.1), multiple PINN instances have to be trained in sequence and/or coupled in parallel on a complex spatio-temporal domain. This usually goes hand in hand with an increase in computational efforts. Furthermore, these methods are ad hoc and generally aim at achieving higher accuracies than the standard approach. This leaves the question of how to choose an appropriate domain decomposition, one that is suitable for overcoming the attractiveness of fixed points in dynamical systems.

With our work, we have contributed to the understanding of how the complexity of the underlying physics loss function and its optimization depend on system settings such as initial values and the computational domain. For certain applications, especially a physics-informed approach with limited labeled training data, modifications to the physics loss function may be necessary in order to maintain the (original) simplicity of PINNs that are applied to complex spatio-temporal domains and systems.

6 Conclusion

In this paper, we studied the physics loss optimization in PINNs when applied to dynamical systems governed by differential equations. The focus on a simple dynamical system revealed that suboptimal solutions corresponding to nonphysical system dynamics can appear as attractive locations in the physics loss landscape. These scenarios potentially disrupt and trap the gradient descent optimization, leading to commonly observed convergence issues in PINNs. We further related the suboptimal solutions in the physics loss landscape to fixed points of the underlying...
physical system which render the minimization of physics loss residuals of the differential equation trivial. Reducing
the computational domain yielded a greater rate of convergence success and, in general, reduced the complexity of
the physics loss optimization. Our insights thus explain the success and superiority of reduced domain methods over
standard PINNs that are optimized on the entire spatio-temporal domain. In the future, we believe that interdisciplinary
research that includes advances in deep learning, stability theory and/or a further understanding of the underlying
physics may improve physics-informed machine learning or benefit from it.

Acknowledgements

The authors acknowledge the financial support of the Austrian COMET — Competence Centers for Excellent Technolo-
gies — Program of the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and
Technology, the Austrian Federal Ministry for Digital and Economic Affairs, and the States of Styria, Upper Austria,
Tyrol, and Vienna for the COMET Centers Know-Center and LEC EvoLET, respectively. The COMET Program is
managed by the Austrian Research Promotion Agency (FFG).

References

[1] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression.
Science Advances, 6(16):eaay2631, 2020.
[2] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse
identification of nonlinear dynamical systems. Proceedings of the national academy of sciences, 113(15):3932–
3937, 2016.
[3] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine
learning–accelerated computational fluid dynamics. Proceedings of the National Academy of Sciences, 118(21),
2021.
[4] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning
to simulate complex physics with graph networks. In International Conference on Machine Learning, pages
8459–8468. PMLR, 2020.
[5] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning
framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of
Computational Physics, 378:686–707, 2019.
[6] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-
informed machine learning. Nature Reviews Physics, 3(6):422–440, 2021.
[7] Ehsan Kharazmi, Min Cai, Xiaoning Zheng, Zhen Zhang, Guang Lin, and George Em Karniadakis. Identifiability
and predictability of integer-and fractional-order epidemiological models using physics-informed neural networks.
Nature Computational Science, 1(11):744–753, 2021.
[8] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed
flows. Computer Methods in Applied Mechanics and Engineering, 360:112789, 2020.
[9] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. The Journal
of Machine Learning Research, 19(1):932–955, 2018.
[10] Minglang Yin, Xiaoning Zheng, Jay D Humphrey, and George Em Karniadakis. Non-invasive inference of
thrombus material properties with physics-informed neural networks. Computer Methods in Applied Mechanics
and Engineering, 375:113603, 2021.
[11] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and
pressure fields from flow visualizations. Science, 367(6481):1026–1030, 2020.
[12] Shengze Cai, Zhicheng Wang, Frederik Fuest, Young Jin Jeon, Callum Gray, and George Em Karniadakis. Flow
over an espresso cup: inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren
via physics-informed neural networks. Journal of Fluid Mechanics, 915, 2021.
[13] Qiming Zhu, Zeliang Liu, and Jinhui Yan. Machine learning for metal additive manufacturing: predicting
temperature and melt pool fluid dynamics using physics-informed neural networks. Computational Mechanics,
67(2):619–635, 2021.
[14] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-
constrained deep learning without simulation data. Computer Methods in Applied Mechanics and Engineering,
361:112732, 2020.
[15] Wei Peng, Weien Zhou, Jun Zhang, and Wen Yao. Accelerating physics-informed neural network training with prior dictionaries. *arXiv preprint arXiv:2004.08151*, 2020.

[16] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1), 2009.

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249--256. JMLR Workshop and Conference Proceedings, 2010.

[18] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055--A3081, 2021.

[19] Franz M Rohrhofer, Stefan Posch, and Bernhard C Geiger. On the Pareto Front of Physics-Informed Neural Networks. *arXiv preprint arXiv:2105.00862*, 2021.

[20] Suryanarayana Maddu, Dominik Sturm, Christian L Müller, and Ivo F Sbalzarini. Inverse Dirichlet Weighting Enables Reliable Training of Physics Informed Neural Networks. *Machine Learning: Science and Technology*, 2021.

[21] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

[22] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 2021.

[23] John Hanna, Jose V Aguado, Sebastien Comas-Cardona, Ramzi Askri, and Domenico Borzacchiello. Residual-based adaptivity for two-phase flow simulation in porous media using Physics-informed Neural Networks. *arXiv preprint arXiv:2109.14290*, 2021.

[24] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

[25] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A*, 476(2239):20200334, 2020.

[26] Cosmin Anitescu, Elena Atroshchenko, Naif Alajlan, and Timon Rabczuk. Artificial neural network methods for the solution of second order boundary value problems. *Computers, Materials and Continua*, 59(1):345--359, 2019.

[27] Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

[28] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[29] Jon A Rivera, Jamie M Taylor, Ángel J Omella, and David Pardo. On quadrature rules for solving Partial Differential Equations using Neural Networks. *arXiv preprint arXiv:2111.00217*, 2021.

[30] Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence and generalization of physics informed neural networks. *arXiv e-prints*, pages arXiv--2004, 2020.

[31] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs. *arXiv preprint arXiv:2006.16144*, 2020.

[32] Olga Fuks and Hamdi A Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020.

[33] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[34] Colby L Wight and Jia Zhao. Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks. *arXiv preprint arXiv:2007.04542*, 2020.

[35] Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002--2041, 2020.
[36] Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021.

[37] Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.

[38] Zheyuan Hu, Ameya D Jagtap, George Em Karniadakis, and Kenji Kawaguchi. When Do Extended Physics-Informed Neural Networks (XPINNs) Improve Generalization? *arXiv preprint arXiv:2109.09444*, 2021.

[39] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18, 2018.

[40] Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.

[41] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *arXiv preprint arXiv:2102.04626*, 2021.

[42] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[43] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[44] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.