Studying the Damped Harmonic Oscillator with Physics Informed Neural Networks

Daniele Montagnani

September 8, 2024

1 Introduction

When trying to learn a new problem solving approach, I prefer to start by seeing how it performs of problems I already understand well enough. In this project I will test the Physics-Informed approach to training neural networks in a simple setting: learning the dynamics of the damped harmonic oscillator. I chose the damped harmonic oscillator because it is a simple, but extremely relevant, dynamical system. Furthermore, its solutions can contain many interesting behaviors (oscillatory, exponential decay) whose nature is best described in the language of differential calculus.

2 Damped Harmonic Oscillator

2.1 Analytical treatment

The damped harmonic oscillator, illustrated for example in Steve Brunton (2023), is a dynamical system used to describe a system formed by a mass m attached to a spring of constant k whose movement is antagonised by a damper with coefficient d. By computing the two forces acting on the mass at every moment, and by applying the second law of dynamics, we find that the dynamics of the system is described by the following differential equation: $m\ddot{x} + d\dot{x} + kx = 0$.

Now, there are different ways to solve the differential equation given above. I choose to perform a reduction of order by introducing an auxiliary variable $v = \dot{x}$. This is to match what has been done in the code: in order to have sharper insights into the behavior of the networks, I had it predict both position and velocity.

Proceeding with the solution, we set up a system of first order differential equations:

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{d}{m} \end{pmatrix}\begin{pmatrix} x \\ v \end{pmatrix}$$

Then, we solve the characteristic equation by looking for the eigenvalues of the differential matrix and obtain: $\lambda_{1,2} = \frac{-\frac{d}{m} \pm \sqrt{\left(\frac{d}{m}\right)^2 - 4\frac{k}{m}}}{2}$. And find the general solution: $x(t) = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t}$.

Depending on the sign of the characteristic equation's discriminant we can obtain three different behaviors: overdamped(positive), critically damped(null), and underdamped(negative). I will consider only the underdamped case as it is the most interesting one for the purposes of the current project.

2.2 Simulation

In the code, I simulated the system by using scipy's odeint. I considered an underdamped setting with parameters: $m = 1, d = 0.1, k = 1$. Furthermore, I set initial conditions as: $x_0 = 0, v_0 = 1$.

The solution was the dynamics depicted in Figure 1.

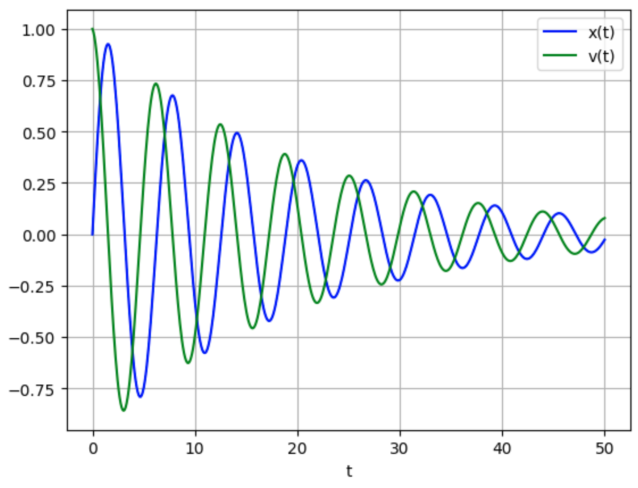

Figure 1: Simulated Dynamics of the System

3 Neural Networks

I trained neural networks on the system dynamics following different approaches: a Traditional Approach, a Pure Physics Informed Approach, and an Hybrid Approach. Training the networks required many trials to find a maximally interesting configuration of "hyper-parameters". In order to perform this exploration I had to define some utility code, for example, to control the position and density of training samples.

3.1 Traditional Approach

Starting with the traditional approach to supervised learning, I trained a traditional (MSE loss between predictions and labels) neural network. Notably, I framed this problem (as well as the physics informed one) as a $\mathbb{R} \to \mathbb{R}^2$ regression where I asked my NN to predict both position and velocity from time. It would have been sufficient to predict only the position, however, I wanted to study also the velocity to get more insight and sanity checks.

3.2 Pure Physics Informed Approach

Coming to the hearth of the project, I trained a Physics Informed Neural Network, explained for example in: Steven Brunton and Kutz (2022), Raissi et al. (2019), Moseley (2021). In the data extraction process, I defined the target gradients $(\dot{x}, \dot{v})$ by transforming the extracted samples with the differential operator matrix. Then, in the training loop, I used pytorch autograd, described in PyTorch (2023), to compute the gradients of the network predictions. I formed a "differential loss" by computing the MSE between the real derivatives of position and velocity and the networks' gradients. Then, I summed to this loss an "initial value loss", computed again as the squared error on initial values.

3.3 Hybrid Approach

Lastly to improve performances, I trained a Physics Informed Neural Network by using an "hybrid loss". The "hybrid loss" is a weighted sum of the "traditional loss", the "differential loss" and the "initial value loss".

3.4 Comparing Performances

Since the setting is simple enough to allow it, I prefer to compare performaces by visually inspecting performances on graphs rather than using accuracy metrics. This approach, albeit less rigorous, will provide more insights on the behavior of the models. In Figure 2, we can see the performances of the different approaches when trained with 5% of the data points. I chose to give my network few data points because PINN's are commonly known to be most useful when data is scarce.

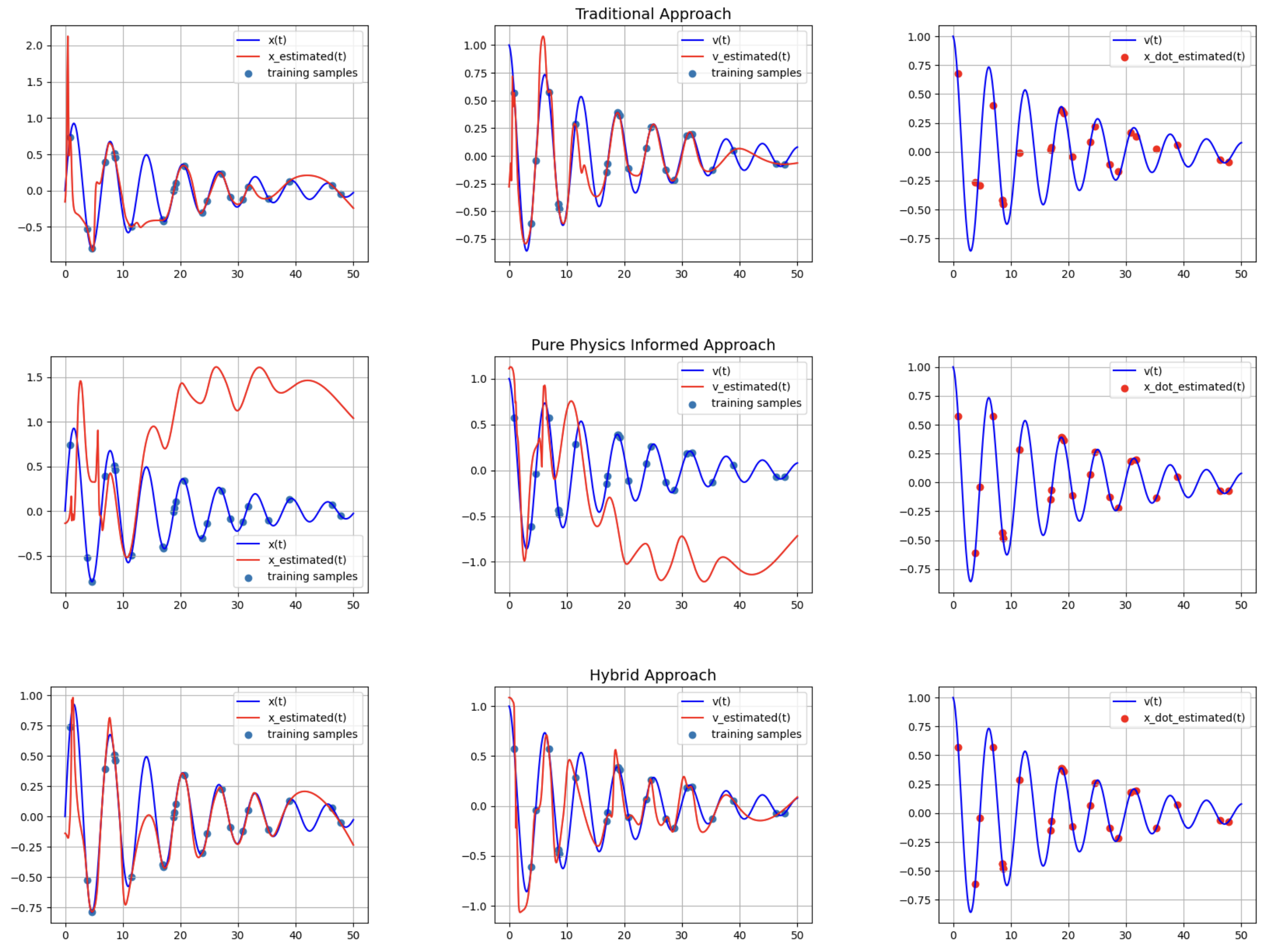

Figure 2: Performances of different approaches, 5% sample

Clearly, in a low data regime, the traditional approach generalises poorly, as it simply tries to intersect the training points. Surprisingly, we can see how the traditional approach approximates the position gradient in a satisfactory way, even though this behavior was not enforced explicitly.

The approach using only the physical loss, on the other hand, performs decently near the initial value but starts overshooting after a sufficient number of gradient errors have been integrated (in analogy with a random walk). In fact, once the estimated function has drifted away, it will continue to approximate gradients, but far away from the real function values. The hybrid approach, as I expected, is the best performer in a low data regime. It shows clear generalisation improvements with respect to the traditional approach and fits the data well.

3.5 Additional Considerations

3.5.1 Appropriate Inductive Bias

In setting up the network architecture I had started with ReLu activation, however, as it can be seen in figure 3, I couldn't bring my PINN up to satifiable performance.

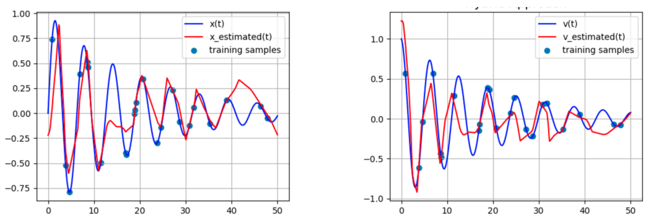

Figure 3: Performance of the hybrid network with ReLu activation

Performances improved vastly by using a highly nonlinear activation such as the tanh. In restrospect, this makes sense as a linear activation function has a poor variety of gradients (the relu has 2 values for its derivative). As explained insightfully in Steve Brunton (2024), particular loss functions and training approaches come paired with appropriate inductive biases, which in turn are determined by the model architecture, Goyal and Bengio, 2020.

3.5.2 Data Availability

In a setting where more data is available, the PINN approach seems to lose its edge wrt the traditional approach. Intuitively, when more data is available the regularisation effect of the differential loss might not be needed and an overall simpler approach might perform better.

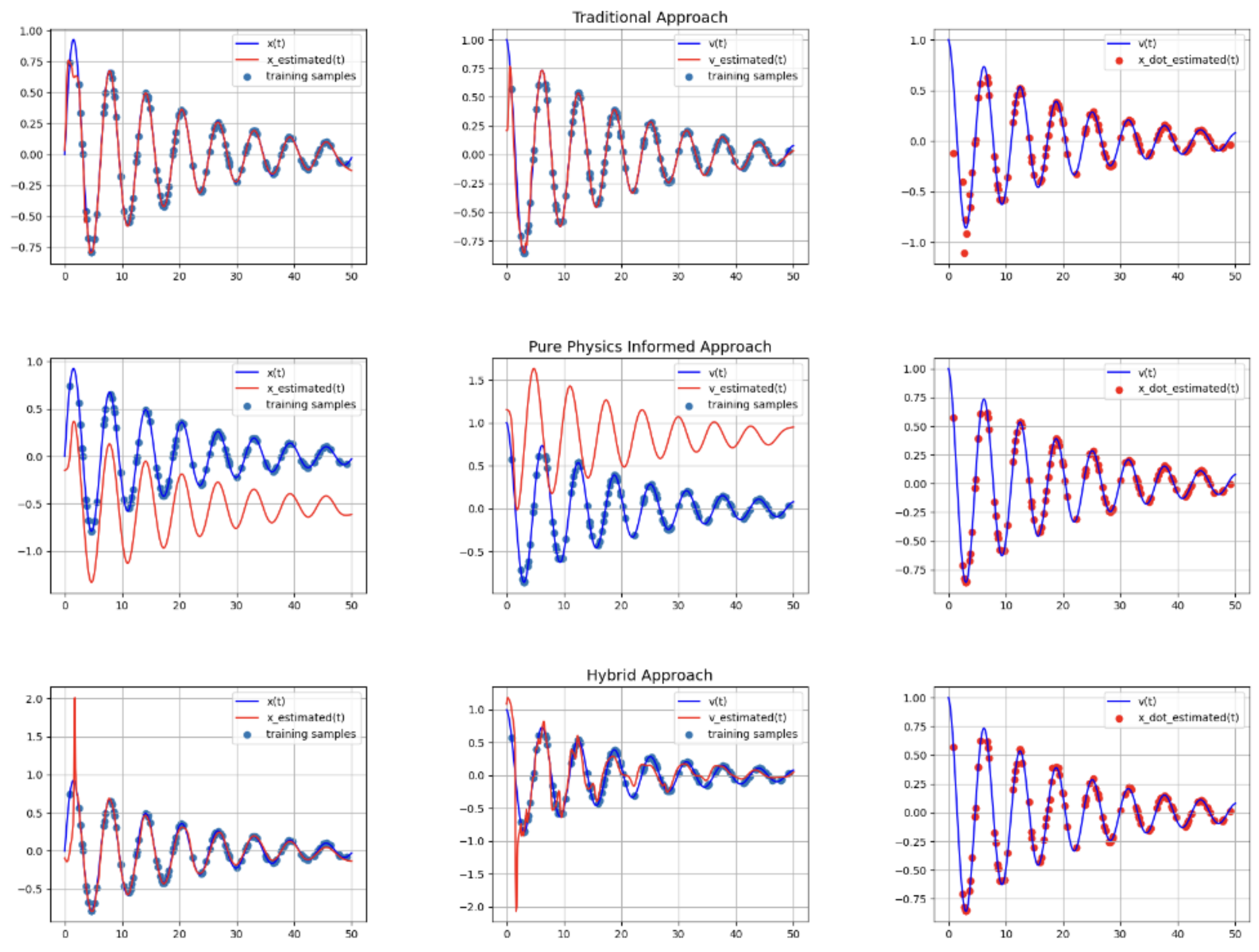

Figure 4: Performances with 25% of the data points available for training

References

Brunton, Steve (2023). *Example Second-Order ODE: Spring-Mass-Damper*. URL: `https://www.youtube.com/watch?v=XXjoh8L1HkE&list=PLMrJAkhIeNNTYaOnVI3QpH7jgULnAmvPA&index=20`.

— (2024). *Physics Informed Machine Learning*. URL: `https://www.youtube.com/playlist?list=PLMrJAkhIeNNQ0BaKuBKY43k4xMo6NSbBa`.

Brunton, Steven and Nathan Kutz (2022). *Data-Driven Science and Engineering*. URL: `http://databookuw.com.`.

Goyal, Anirudh and Yoshua Bengio (Nov. 2020). "Inductive Biases for Deep Learning of Higher-Level Cognition". In: URL: `http://arxiv.org/abs/2011.15091`.

Moseley, Ben (2021). *So, what is a physics-informed neural network?* URL: `https://benmoseley.blog/my-research/so-what-is-a-physics-informed-neural-network/`.

PyTorch (2023). *Automatic differentiation package - torch.autograd*. URL: `https://pytorch.org/docs/stable/autograd.html`.

Raissi, M., P. Perdikaris, and G. E. Karniadakis (Feb. 2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707. ISSN: 10902716. DOI: `10.1016/j.jcp.2018.10.045`.