# Physics-Informed Neural Networks for Data-Efficient Learning


Shaan Ajay Desai

Somerville College

University of Oxford

A thesis submitted for the degree of

*Doctor of Philosophy*

Michaelmas 2021

# Acknowledgements

The PhD journey has been as much about personal growth as it has been about reflecting on the kind, thoughtful and important people in my life. I feel deeply grateful and blessed for everyone who has made it such a special journey. I'd like to begin by thanking my parents who have always supported and championed me to dream big - this is our success. I'd like to give special thanks to Stephen Roberts and Marios Mattheakis for sharing their boundless love for science with me - I was extremely lucky to have you both as mentors on this deeply challenging yet fulfilling road to pushing the frontiers of ML and physics. There are so many Oxford people to thank, but a special thanks goes to Michael, Ondine, Jesse, Laura and Mees - you were the only warmth in my time at Oxford! To my Jaho friends, Kenna, Carol and Maria - who would have thought that such a fortuitous encounter in a small cafe could result in such a strong bond! As always a special thanks to Eshaan and Will - not enough words to describe how lucky I have been to have you both by my side! The biggest thanks undoubtedly goes to Mark James Wood - this PhD was only special because of you. Thank you for being the brother I never had, for your encouragement, thoughtfulness and for selflessly picking up the phone everyday to talk. The 2am walks home in your city, or walking up to a stranger in Harvard yard, and even your political banter. I cannot wait for us to be in NY together, as always, Marks and Sparks! Finally, I would like to thank the Rhodes Trust for their generous support during my time as a PhD Student.

# Abstract

The physical world around us is profoundly complex and for centuries we have sought to develop a deeper understanding of how it functions. Building models capable of forecasting the long term dynamics of multi-physics systems such as complex blood flow, chaotic oscillators and quantum mechanical systems thus continues to be a critical challenge within the sciences. While traditional and computational tools have dramatically improved to address parts of this open problem, they face numerous challenges, remain computationally resource intensive, and are susceptible to severe error accumulation. Now, modern machine learning techniques, augmented by a plethora of sensor data, are driving significant progress in this direction, helping us uncover sophisticated relationships from underlying physical processes. An emergent area within this domain is hybrid physics-informed machine learning where partial prior knowledge of the physical system is integrated into the machine learning pipeline to improve predictive performance and data-efficiency. In this thesis, we investigate how existing knowledge about the physical world can be used to improve and augment predictive performance of neural networks. First, we show that learning biases designed to preserve structure, connectivity and energy such as graphs, integrators and Hamiltonians can be effectively combined to learn the dynamics of complex many-body energy-conserving systems from sparse, noisy data. Secondly, by embedding a generalized formalism of port-Hamiltonians into neural networks, we accurately recover the dynamics of irreversible physical systems from data. Furthermore, we highlight how our models, by design, can discover the underlying force and damping terms from sparse data as well as reconstruct the Poincaré section of chaotic systems. Lastly, we show that physics-informed neural networks can be effectively exploited for efficient and accurate transfer learning - achieving orders of magnitude speed-up while maintaining high-fidelity on numerous well studied differential equations. Collectively, these innovations show case a new direction for scientific machine learning - one where existing knowledge is combined with machine learning methods. Many benefits naturally arise as a consequence of this including (1) accurate learning and long-term predictions (2) data-efficiency (3) reliability and (4) scalability. Such hybrid models are paramount to developing robust machine learning methods capable of modeling and forecasting complex multi-fidelity, multi-scale physical processes.

# Declaration

I declare that this thesis is my own work and it is submitted for the degree of Doctor of Philosophy at the University of Oxford. No part of this thesis has been submitted for any other qualification. A summary of publications is provided below. Published material that appears in the thesis is marked by * and those that do not are marked by **.

**Variational Integrator Graph Networks for Learning Energy Conserving Dynamical Systems*** (Chapter 3 of thesis)

- Publication Status: Published
- Publication Details: Shaan A. Desai, Marios Mattheakis, and Stephen J. Roberts. Variational Integrator Graph Networks for Learning Energy Conserving Dynamical Systems. Phys. Rev. E 104, 035310, 30 September 2021.
- Paper Contributions: Shaan conceived the idea and wrote the code. Shaan, Marios and Stephen wrote the paper and analysed the results.

**Port-Hamiltonian Neural Networks for Learning Explicit Time-Dependent Dynamical Systems*** (Chapter 4 of thesis)

- Publication Status: Published
- Publication Details: Shaan A. Desai, Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen J. Roberts. Phys. Rev. E 104, 034312. 29 September 2021.
- Paper Contributions: Shaan conceived the idea and wrote the code. Shaan and Marios analysed the results and designed the experiments. David proposed physical systems to investigate. Pavlos and Stephen made edits to the paper.

**One-Shot Transfer Learning of Physics-Informed Neural Networks*** (Chapter 5 of thesis)

- Publication Status: Submitted for Publication

- Publication Details: Shaan A. Desai, Marios Mattheakis, Hayden Joy, Pavlos Protopapas, and Stephen J. Roberts. AISTATS [under review]
- Paper Contributions: Shaan developed the idea for transfer learning PINNs. Shaan and Marios designed the experiments. Shaan wrote the code. Hayden ran a set of experiments. Shaan, Marios and Stephen wrote the paper. Pavlos first proposed the idea of transfer learning of differential equations and made edits to the paper.

**DYNOTEARS: Structure Learning from Time-Series Data**[**]

- Publication Status: Published
- Publication Details: Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, Bryon Aragam. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR 108:1595-1605, 2020.
- Paper Contributions: Philip and Konstantinos conceived the idea. Roxana and Nisara formalized the proofs. Shaan, Roxana and Nisara ran experiments. Everyone contributed to authoring and editing the paper.

**Tuning Mixed Input Hyperparameters on the Fly for Efficient Population Based AutoRL**[**]

- Publication Status: Published
- Publication Details: Jack Parker-Holder, Vu Nguyen, Shaan Desai, Stephen J. Roberts. NeurIPS 2021.
- Paper Contributions: Jack conceived the idea. Vu proved theoretical bounds for the method. Shaan and Jack ran experiments. Jack and Steve wrote the paper.

# Contents

*What is man, that though art mindful of him*

— Psalms 8:4-8

# 1 Introduction

## Contents



## 1.1 Motivation

The physical world around us is profoundly complex and for centuries scientists have sought to develop a better understanding of how it functions. Building models that can accurately capture and forecast the dynamics of physical systems therefore lies at the cornerstone of scientific progress as such models have helped us discover new materials (Rhone et al. 2020), understand blood flow in the human body (Melchionna et al. 2011), and develop a deeper understanding of human preferences in economies (Achdou et al. 2014). While traditional mathematical and computational tools have dramatically improved to meet the growing need to address this open problem, they face numerous challenges. In particular, they remain computationally resource intensive, are susceptible to severe error accumulation and require complicated formulations and code. Now, modern machine learning techniques, augmented by a plethora of data, are able to help us uncover sophisticated relationships from

underlying physical processes without the bottlenecks of traditional approaches. An emergent area within this domain is hybrid physics-informed machine learning in which partial prior knowledge of the physical system is integrated into the machine learning pipeline to improve predictive performance, data-efficiency and interpretability. While still in its infancy, research in this direction has already demonstrated significant breakthroughs at modeling complex fluid motion (Sanchez-Gonzalez, Godwin, et al. 2020), predicting high-dimensional partial differential equations (Karniadakis et al. 2021) and in designing fingerprints for molecular dynamics (Z. Li et al. 2021).

Modelling complex physical phenomena from multi-modal (e.g. visual, textual and state) and multi-fidelity data requires robust machine learning (ML) methods that are scalable. While many ML methods exist to tackle such problems, Neural Networks (NNs) have demonstrated remarkable promise in their ability to learn complex relationships from multiple sources of data. For example, they have been successfully leveraged to classify objects in images (He, Gkioxari, et al. 2017), translate languages (Devlin et al. 2019), and even learn phase transitions in materials (V. Bapst et al. 2020). These successes have made NNs a go-to function approximator for many challenging data-driven problems. We must note that alternative approaches such as kernel methods are equally capable of learning such problems, albeit, with their own benefits and limitations compared to NNs. This thesis focuses on an emergent area within machine learning, namely, hybrid physics-informed machine learning, where prior knowledge of the underlying system is embedded into the model to enable better representations and improve generalisation performance. For many of the systems investigated in recent literature, inductive biases and constraints have come into NNs seamlessly, for example by capturing relational structure via graph neural networks, or by recasting the problem in terms of meta-objects like the Hamiltonian in the computational graph. These advances make it easy to stitch together inductive biases in NNs, as we later show in Ch.3 and Ch.4. While some work has been done to embed priors into kernels (Rath et al. 2021) there is limited research that explores how to combine these priors. For

kernel methods, combinations of priors might invoke the use of hierarchical (deep) GPs that are notoriously fragile (Dunlop et al. 2018). While kernel approaches do allow us to capture prior information about the function space, NN approaches can be augmented via Gram-Schmidt orthogonalisation, for example, to enforce better representations. For these reasons, we investigate NNs as the function approximator of choice in this thesis, but their use does not preclude future use of other approaches should some of the above difficulties associated with (e.g.) constrained, deep Gaussian Processes prove to be resolved in a tractable manner.

In this thesis, we investigate how existing knowledge about the physical world, commonly referred to as an inductive or learning bias, can be used to improve and augment the predictive performance of neural networks along three main directions. First, by conducting an extensive ablation across recent advances, we show that learning biases capable of preserving structure, connectivity and energy such as graphs, integrators and Hamiltonians can be effectively combined and embedded in a neural network to learn the dynamics of complex many-body energy-conserving systems from sparse, noisy data. Learning such energy-conserving systems efficiently from data is relevant across a host of domains including molecular dynamics, chemical reactions, planetary motion and robotics. Secondly, by embedding a generalized formalism of port-Hamiltonians into neural networks, we demonstrate that the dynamics of irreversible physical systems that undergo forcing and damping can be accurately recovered from limited data. Being able to recover the physical quantities, and thus forecast the motion of damped and driven systems is relevant to robotics, weather forecasting, mechanical and industrial engineering and many other domains. Lastly, we show that physics-informed neural networks - networks capable of satisfying known differential equations, can be effectively exploited for efficient and accurate transfer learning, achieving orders of magnitude speed-up while maintaining high-fidelity on numerous well studied differential equations. Transfer learning using these networks is paramount, speeding up innovation and identifying solutions under new conditions without having to retrain an entire network. Collectively, these innovations show a new direction for scientific machine

learning - one where existing knowledge is combined with data-driven predictive techniques. Many benefits naturally arise as a consequence of this including (1) accurate learning and long-term predictions (2) data-efficiency (3) reliability and (4) scalability. Such hybrid models are key to building robust machine learning methods capable of modeling and forecasting complex multi-fidelity, multi-scale physical processes. Indeed they have already been shown to outperform classical approaches in a range of settings and are likely to continue on this path to become a dominant approach in scientific machine learning.

## 1.2 Contribution

In this thesis, we make three major contributions to the existing literature. We summarize these advances and the major outcomes from our investigation of these systems below.

- Variational Integrator Graph Networks - by conducting a large ablation study across all the recent innovations in physics-informed priors, we identify a new class of unexplored networks we refer to as variational integrator graph networks, capable of learning the underlying dynamics of many-body systems from sparse noisy data (code can be found here [1]). The core outcomes from this work are:
  - We unpack recent innovations by grouping their inductive biases into generalized segments.
  - We then systematically investigate all possible combinations of these individual segments.
  - In doing so we generalize existing approaches, and identify a new class of networks we term Variational Integrator Graph Networks (VIGNs).
  - Theoretically, these networks combine graphs, symplectic integrators and energy constraints together, making them robust in numerous settings.

[1] https://github.com/shaandesai1/VIGN

  - In particular, the ablation study helps highlight which biases are crucial under specific settings.

- Port-Hamiltonian Neural Networks - we embed a mathematical formalism used to describe energy preserving systems undergoing systematic damping and forcing through time into a neural network to improve learning of irreversible systems from data (code can be found here [2]). The core outcomes from this research are:

  - We adopt a port-Hamiltonian formalism to learn irreversible physical processes for which energy is changing in time.
  - We demonstrate that our network can efficiently learn the dynamics of nonlinear physical systems of practical interest, while accurately recovering the underlying stationary Hamiltonian, time-dependent force and dissipative coefficient, exclusively from position and momentum data.
  - A unique outcome of this is our ability to use the model to accurately recover the Poincaré section of a chaotic Duffing oscillator.
  - The approach is fundamental to learning non-autonomous systems from data and is of practical interest e.g. to the robotics community.
  - Using the recovered parameters, the model can be investigated under different driving and dissipative conditions.

- One-Shot Transfer Learning of Physics-Informed Neural Networks - we show that a physics-informed network can be batch trained on numerous differential equations simultaneously and then leveraged to instantly identify solutions to other, similar differential equations (code can be found here[3]). Notable outcomes from this work are:

[2] https://github.com/shaandesai1/PortHNN
[3] https://github.com/shaandesai1/TransferDE

- A model can be trained to satisfy multiple equations at once enabling us to learn a rich latent space, ripe for transfer learning.
- For linear systems, fine-tuning the output layer for new differential equations is equivalent to solving the normal equations.
- Linear systems can therefore be solved in one-shot, eliminating the need for retraining an entire network when the differential equation changes.
- Fine-tuning the last layer in the network typically leads to a dramatic improvement in performance even when the initial hidden states are minimally trained.
- The network can also be used to solve non-linear equations, in which case an optimization scheme such as gradient descent may be used to train the model.
- The one-shot learning paradigm results in rapid inference and high-fidelity solutions across multiple ODEs and PDEs.

In the chapters that follow, each of the contributions listed is presented in a self-contained manner. This is done so that the reader can refer to a specific chapter and understand the core concepts.

*If I have seen further it is by standing on the shoulders of Giants.*

— Isaac Newton

# 2 Background

## Contents



## 2.1 Preliminaries

Mathematical and computational tools have long been used to develop models for dynamical systems. Now, modern machine learning techniques such as neural

networks, capable of learning complex relationships from vasts amounts of data, are paving a new pathway to model dynamical systems. Despite extensive progress in this direction, most successful neural network models remain black-boxes i.e. are uninterpretable. An emergent approach to tackle this problem is to combine traditional bottom up mathematical modeling with machine learning. Specifically, it has been extensively shown that prior physics knowledge embedded into neural networks not only improves their interpretability but also significantly improves their predictive performance. Here, we conduct an extensive literature review of recent methods that exploit prior physics knowledge to learn dynamical systems. We explicate the major advances, focusing on neural network innovations, that are pushing the frontier and highlight gaps that we later address in the thesis.

We provide a brief summary of the essential components used to describe dynamical systems before detailing the major innovations in recent literature.

### 2.1.1 Differential Equations

Traditionally, a dynamical system can be described by a differential equation (DE) that governs the rate of change of variables over time. We begin by describing a general first-order ordinary differential equation (ODE) of the form:

$$\dot{\mathbf{s}} = f(\mathbf{s}, t); \quad \dot{\mathbf{s}} = \frac{\mathrm{d}\mathbf{s}}{\mathrm{d}t}, \tag{2.1}$$

where $\mathbf{s}(t)$ is a time-dependent state vector i.e. $\mathbf{s}(t) = [s_1(t), s_2(t), ...., s_n(t)]$, $n$ is the number of variables and $f$ is a linear or non-linear function of the states and time $t$. Such equations are ubiquitous in understanding and forecasting the dynamics of physical processes and form the basis for much of the work in this thesis. For many physical systems, the state vector comprises of $\mathbf{s}(t) = [\mathbf{q}(t), \mathbf{p}(t)]$, where $\mathbf{q}$ is a time-dependent position vector and $\mathbf{p}$ is a time-dependent momentum vector. If the full functional form of the equation for $f$ is known a priori, as well as the initial condition $\mathbf{s}_0$, it is possible to integrate the differential equation and identify a solution at various points in time as follows:

$$\mathbf{s}(t_1) = \mathbf{s}_0 + \int_{t_0}^{t_1} f((s), t)\mathrm{d}t, \tag{2.2}$$

where $t_0$ is the initial time and $t_1$ the final time. In some special cases, this can be done analytically. However, many complex physical systems are analytically intractable, requiring numerical methods, such as discrete integrators like Runge-Kutta, to be used to approximate the solution (see Appendix A). However, new machine learning methods, which we later describe, provide an alternate route to solving such problems without the bottlenecks faced by traditional integrative approaches. In addition, if the functional form of $f$ is not known, then it needs to be approximated from available data. Many time-series methods exist to identify the underlying form of $f$ but we later show how neural networks combined with different physics priors can be used to improve the approximation of $f$.

Although we only specify a general first-order ODE here, such equations can be written for higher-order derivatives as well as for more variables e.g. $\mathbf{s}(t, x, y..)$ - such systems are known as Partial Differential Equations (PDEs). We later show, in Chapter 5, how such systems can be solved using neural networks.

### 2.1.2 Hamiltonian Mechanics

Some ordinary differential equations that describe the dynamics of energy conserving physical phenomena can be described by Hamiltonian mechanics (Marsden et al. 2001). The Hamiltonian is an important representation of a dynamical system because it generalizes classical mechanics. The Hamiltonian $\mathcal{H}$ is a scalar function of position $\mathbf{q} = (q_1, q_2, ...., q_M)$ and momentum $\mathbf{p} = (p_1, p_2, ...., p_M)$ and is formalized by Hamilton's equations:

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}. \tag{2.3}$$

This formalism shows that the time derivatives can be obtained by taking partial derivatives of the Hamiltonian with respect to the position and momentum vector. Furthermore, it can be shown that the gradient induced by Hamilton's equations is symplectic, meaning that a transition along any direction of the gradient preserves the scalar Hamiltonian $\mathcal{H}$. For many physical systems, the Hamiltonian also reflects the total energy of the system which implies that moving along the symplectic

gradient preserves the energy of the system. Therefore, the Hamiltonian is a powerful inductive bias that can be utilized to evolve a physical state while maintaining energy conservation. Since many well-known classical mechanical systems can be described in this way, we later identify networks that embed this formalism and are thus capable of learning from sparse, noisy data.

### 2.1.3 Lagrangian Mechanics

Lagrangian mechanics offers an alternative to the Hamiltonian for generalizing a dynamical system. Rather than position and momentum (canonical coordinates) defining the state space, Lagrangian mechanics is defined using a generalized coordinate state space $(\mathbf{q}, \dot{\mathbf{q}})$. This is particularly useful in physical settings where the description and measurement of generalized coordinates may be easier to work with than canonical coordinates (Marsden et al. 2001). Given these coordinates, Joseph-Louis Lagrange (Lagrange 1811) showed that a scalar value $\mathcal{A}$, referred to as the action, can be defined as the integral of a Lagrangian, $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$:

$$\mathcal{A} = \int_t^{t+1} \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) \mathrm{d}t \tag{2.4}$$

The integral can be thought as inducing multiple paths between points in state space i.e. multiple walks in the domain of $(\mathbf{q}, \dot{\mathbf{q}})$. However, only one path is a stationary state of the action integral. This state lets us move from $t \to t+1$ with minimal energy. It can be shown, through variational calculus, that this stationary state must satisfy the Euler-Lagrange equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) = \frac{\partial \mathcal{L}}{\partial \mathbf{q}}. \tag{2.5}$$

It is indeed possible to show that there is a transformation from the Euler-Lagrange equations to the Hamiltonian equations. Another unique outcome of this formalism is that it connects tightly to variational calculus which can describe new integrators capable of preserving energy (see Appendix.A.4 for details). We later describe systems where the Lagrangian is embedded into neural networks.

### 2.1.4 Neural Networks

Multilayer Perceptrons (MLPs), the simplest Neural Networks (NNs), are provable universal function approximators (Hornik et al. 1989). As such, they have been used as a foundation to design complex models, which have drawn significant attention over the past decade, capable of solving difficult data-driven problems, from image classification (He, Gkioxari, et al. 2017; He, X. Zhang, et al. 2016) to speech-to-text tasks (Devlin et al. 2019). Here, we briefly describe the core components of neural networks.

In a feed-forward neural network, each layer consists of a number of units (neurons), that first computes a weighted linear combination, $\mathbf{z} = \mathbf{W}^T\mathbf{x} + \mathbf{b}$, of the layer input, $\mathbf{x}$, followed by an element-wise non-linearity, $\sigma(\mathbf{z})$ (see Fig.2.1). The model parameters consist of weights $\mathbf{W}$ and bias terms $\mathbf{b}$ that need to be estimated via data-driven training. Further, the non-linear function, $\sigma$, needs to be chosen (though for many of the internal 'fully-connected' layers of the network, this non-linearity is a smooth squashing function, often $\tanh(\cdot)$). Such transformations can be repeatedly chained, with varying nonlinear link functions and differing numbers of neural units across layers, to yield an output $\mathbf{y}$. Such multiply-layered models are known as Deep Neural Networks (DNNs). To increase their expressiveness and induce richer latent representations, modern DNNs include additional features such as compressive layers (e.g. pooling layers), batch normalization, and Rectified Linear Unit (ReLU) activation functions (Goodfellow et al. 2016).

In general, all neural networks operate by assuming that the layer transformations can be chained, with each layer being a (nonlinear) function of the one that preceded it. For the first hidden layer $\mathbf{z}_1$, with network inputs $\mathbf{x}$, we may thus write:

$$\mathbf{z}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \tag{2.6}$$

and for the n'th hidden layer:

$$\mathbf{z}_n = \sigma(\mathbf{W}_n\mathbf{z}_{n-1} + \mathbf{b}_n). \tag{2.7}$$

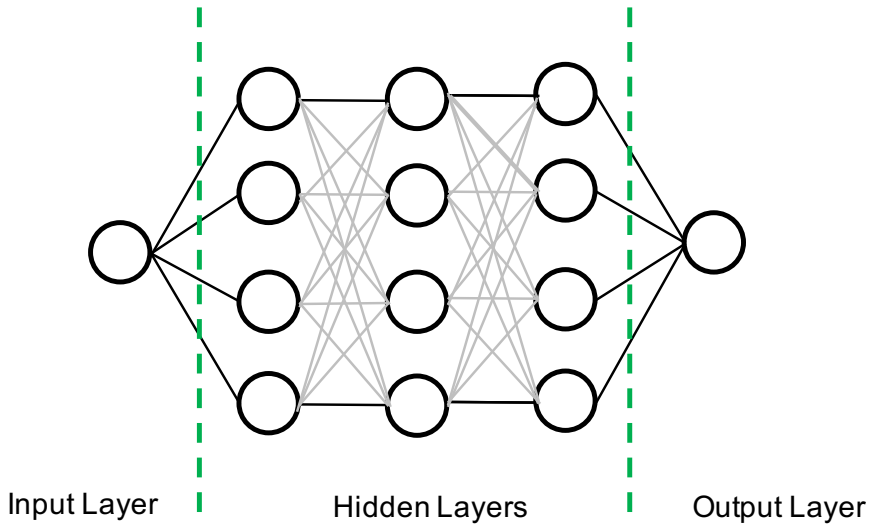



**Figure 2.1:** Neural Networks transform inputs to outputs. The example network has a single input and output layer, and three hidden layers. Each hidden layer consists of 4 neurons such that each neuron transforms the previous layer's outputs via a function defined via a set of parametric weights and bias terms. Intermediary transformations are referred to as hidden layers.

To train the network, a loss function is defined associated with the output and a target variable. The goal of learning is to therefore optimize the parameters $\{\mathbf{W}, \mathbf{b}\}$ of the network with respect to the loss function. Although there are many optimizers that could be used, the sheer scale of most modern neural networks precludes the use of second-order methods and is reliant on modifications of first-order approaches, such as (stochastic) gradient descent (or a variant such as Adam (Kingma et al. 2015)). Using such an optimization scheme requires computing gradients of the loss function with respect to each of the parameters of the network. To this end, a computationally efficient mechanism that exploits the chain-rule, known as Autograd (Maclaurin et al. 2015), is typically leveraged to automatically compute gradients. As we will show, this gradient computation method is extensively used to enforce physics based constraints.

### 2.1.5 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) (Rumelhart et al. 1986) are often used for sequence modeling. They leverage the power of deep networks, but are able to preserve the time-dependence that exists between data-points via a buffer of

hidden states that memorize information from the past while recursively being updated by current information. Given time-series data $\mathbf{x}_{1:T}$, a recurrent neural network is typically defined as:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{z}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}) \tag{2.8}$$

where $\mathbf{W}_z$ and $\mathbf{W}_x$ are the hidden and input weights respectively, $\mathbf{b}$ is the bias and $\sigma$ is the non-linearity. By computing all the hidden states $\mathbf{z}$, a loss function can be defined and then optimized. Indeed the recursive operations lead to a complex chain rule that can result in the vanishing gradient problem - when gradients of the weights near the output of the network are small, the chain rule will induce even smaller gradients for weights near the input, resulting in no updates. By connecting ordinary differential equations to recurrent neural networks, (R. T. Q. Chen, Rubanova, et al. 2018) show that it is possible to circumvent this issue as we later discuss. Indeed many new extensions to RNNs have been proposed. Arguably the most well known are long-short term memory (LSTM) (Hochreiter et al. 1997) networks that contain additional logic gates between transformations designed to retain some information but also forget information. Recurrent structures are thus fundamental to uncovering time-series dynamics from data.

### 2.1.6 Graph Networks

Thus far we have noted that deep neural networks are capable of transforming inputs to outputs via parametric weights and nonlinear link functions. By incorporating memory units into the network and allowing recurrent updates in the learning procedure, we can capture and preserve temporal structure using recurrent neural networks. In this section, we briefly outline how *graph networks* may be used to preserve relational structures, exploiting message passing on graphs. In line with the work of (Battaglia, Hamrick, et al. 2018), we define a graph as a tuple $(\mathbf{u}, V, E)$ where $(V = \{\mathbf{v}_i\}_{i=1:N^v})$ is the set of graph nodes (of cardinality $N^v$), $(E = \{\mathbf{e}_k, r_k, s_k\}_{k=1:N^e})$ is the set of edges (of cardinality $N^e$) where $\mathbf{e}_k$ is an edge attribute, $r$ is the index of the receiver node and $s$ is the index of the sender node

and ($\mathbf{u}$) is a vector of global graph attributes. One step of a graph network forward pass consists of three update functions and three aggregations. Formally:

$$\mathbf{e}'_k = \theta^e(\mathbf{e}_k, \mathbf{v}_{rk}, \mathbf{v}_{sk}, \mathbf{u}) \quad \bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i) \tag{2.9}$$

$$\mathbf{v}'_i = \theta^v(\mathbf{v}_i, \bar{\mathbf{e}}_i, \mathbf{u}) \quad \bar{\mathbf{e}}' = \rho^{e\to u}(E') \tag{2.10}$$

$$\mathbf{u}' = \theta^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}) \quad \bar{\mathbf{v}}' = \rho^{v\to u}(V') \tag{2.11}$$

where $E'_i = \{\mathbf{e}'_k, r_k, s_k\}_{r_k=i,k=1:N^e}$, $V' = \{\mathbf{v}'_i\}_{i=1:N^v}$ and $E' = \cup_i E' = \{\mathbf{e}'_k, r_k, s_k\}_{k=1:N^e}$. $\theta$'s represent neural networks, each parametrized by their own set of weights. Here, $\rho$ represents an aggregation mechanism, typically element-wise summation, or a mean or max function. First, the network $\theta^e$ computes updated edge attributes $\mathbf{e}'_k$ which are then aggregated per node using $\rho^{e\to v}$. Then, using the aggregated edges $\bar{\mathbf{e}}'$ per node, the network $\theta^v$ updates the node attributes. These node attributes are then aggregated using $\rho^{e\to u}$ to update the global parameter $u'$. As such, graph networks are composed of three networks and three aggregations. Collectively, they update graph attributes and can therefore be chained to form multiple graph transformations. Optimization of such networks exploits the back-propagation of error signals, in the same way as other neural network learning mechanisms and once more can be achieved via (stochastic) gradient methods. We note that each network, $\theta$, uses information about the edges, nodes and global parameters. As can be seen, graph networks are capable of preserving structural and relational data while updating information on a graph. They are a crucial building block in learning dynamical systems described by interacting particles (Battaglia, Hamrick, et al. 2018).

## 2.2 Literature Review

Broadly, the intersection of physics and AI falls into one of two domains, physics for AI or AI for physics. The former uses techniques from physics to develop and improve learning algorithms in general. The latter uses existing learning approaches (with adaptations) to perform inference over physical systems, for example ML for materials (Rupp et al. 2012; Smith et al. 2017). In this review, we focus our

attention on the latter as the three main contributions later presented leverage ML techniques for a range of scientific problems.

Physicists have long been interested in using learning tools to predict physics based systems. Some of these include predicting the magnetic properties of 2-D materials (Rhone et al. 2020), predicting the time evolution of N-body systems (Battaglia, Hamrick, et al. 2018) and even using AI to understand phase transitions (V. Bapst et al. 2020). However, numerous challenges still remain: 1. data-efficient learning, 2. reducing computational cost, 3. improving predictive accuracy and 4. learning better representations of the underlying physical process. Researchers have identified methods to address all these challenges, but arguably the most promising approach hinges on physics-informed priors embedded in learning. It has been shown that models enriched with physically-informed priors i.e. models which consist of some knowledge about the physical system a priori, significantly outperform existing methods in terms of data-efficiency and predictive accuracy. This has sparked a sharp interest in building both task-specific and general physics priors to improve learning. In this section, we summarize some of the core developments in physics informed inductive biases. It is of significance to remind oneself that existing literature in this domain is not deeply interdisciplinary, in that physics, robotics, control and ML have for a long time existed in silos. Whilst developments in each area have been very promising they can benefit from cross-pollination. As we show in this thesis, it is in fact possible to take a data-driven approach from machine learning, embed a physics-informed prior, such as the port-Hamiltonian, and model a system like the Duffing oscillator from control literature. That is to say, by leveraging core principles in multiple fields, it is possible to tackle problems in distinctively new ways.

### 2.2.1 Early Work

Embedding physics informed priors in neural networks dates back over two decades to the seminal works of (Lagaris et al. 1998) and (Howse et al. 1996). (Lagaris et al. 1998) was the first to introduce a novel approach to learn the solutions of known differential equations using neural networks. The authors of the paper propose to

compute analytic partial derivatives of the neural network output solution with respect to the inputs of the network. Using such a framework, they show it is possible to constrain a neural network loss to satisfy a given differential equation under a set of initial and boundary conditions. Arguably, this work has laid the foundation for some of the most novel physics informed machine learning approaches which have emerged in the past few years such as Physics-Informed Neural Networks (PINNs)(Maziar Raissi et al. 2017). Furthermore, in 1996, (Howse et al. 1996) highlighted an approach to identifying dynamical systems from data using physics constraints. In their work, the authors define a generalized functional form for dynamical systems that casts the dynamic problem into a conservative Hamiltonian constraint and non-conservative constraint. Using such a formalism, the authors show their ability to recover parameters of a system more accurately from data. In principle, this work was the first to illustrate how a neural network with energy-based constraints could improve learning.

In addition, the notion of embedding physics-informed inductive biases in neural networks can be found in other domains as well (Rupp et al. 2012; Smith et al. 2017; Witkoskie et al. 2005; Pukrittayakamee et al. 2009; Yao et al. 2018). For example, early efforts (Witkoskie et al. 2005) demonstrate that, in contrast to directly learning a potential energy surface, including an optimization of the gradients of the potential surface can improve performance. This addition means that the learning process can be supplemented with additional information about the underlying physics and can hence improve the learnt potential surface. The fundamental idea being that if we have access to supplemental data such as the gradients, but fewer data points, we might actually learn a surface with higher accuracy than if we had many data points with no gradient information.

These ideas have been essential in developing new machine learning techniques capable of solving dynamical systems.

### 2.2.2 Graph Neural Networks

Earlier, we introduced the notion of graph neural networks and argued that they are designed to preserve relational structure. One application where this is pertinent is in interacting particle systems. Interacting particle systems are inherently well structured since their dynamics can be formulated as a sum of particle to particle interactions. Given this, their dynamics can be represented on a graph $G = (u, V, E)$ (Battaglia, Hamrick, et al. 2018). For example, a node ($V$) can be a particle in a many-body problem. These nodes can be used to represent the core features of the particle like its position, momentum, mass, and other particle constants. Edges ($E$) can represent forces between the particles, and the 'Globals' ($u$) can represent universal constants such as air density, the gravitational constant etc. By exploiting graph neural networks on data structured in this way, we are able to preserve the relational bias that exists between these particles during the training process (Sanchez-Gonzalez, Godwin, et al. 2020; Battaglia, Hamrick, et al. 2018; Battaglia, Pascanu, et al. 2016; Sanchez-Gonzalez, Heess, et al. 2018; Seo and Liu 2019; Cranmer, Sanchez Gonzalez, et al. 2020; Seo, Meng, et al. 2020; Lamb et al. 2020; Cranmer, Sam Greydanus, et al. 2020).

Graph neural networks carry out a sequence of transformations to the graph nodes and edges to update the graph parameters. The representation is therefore powerful for many-body systems primarily because the graph networks can operate within the known constraints of physical systems.

Some major applications of GNNs include:

- GNNs trained to learn Hamiltonians achieve state-of-the-art results in rolling out trajectories of large N-body systems (Sanchez-Gonzalez, Victor Bapst, et al. 2019).
- GNNs have shown significant promise in explaining the phase transitions of glassy materials (V. Bapst et al. 2020).
- GNNs can learn complex fluid like systems from visual data (Sanchez-Gonzalez, Godwin, et al. 2020).

With promising results, GNNs have attracted significant attention in solving a range of physical systems. Indeed further work in this direction stands to investigate the accuracy of the learnt representations, the connection to graph theoretic operators as well as solving the computational challenges of large many-body systems.

### 2.2.3 Neural Ordinary Differential Equations

As was discussed earlier, recurrent neural networks are able to model complex time-series data by sequentially transforming a hidden state. Such sequential transformations also occur in residual networks (He, X. Zhang, et al. 2015) and normalizing flows (Kobyzev et al. 2020). Mathematically, these transformations can be formulated as:

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f_\theta(\mathbf{z}_t), \tag{2.12}$$

where $\mathbf{z}_t$ is a hidden layer of the network and $f_\theta(\mathbf{z}_t)$ is a neural network parametrized by $\theta$ that transforms the hidden layer. It can be shown that the sequential updates are equivalent to an Euler discretization of a continuous ODE (Y. Lu et al. 2017). In the continuous limit of any of these time series based networks, Equation 2.12 becomes:

$$\frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} = f(\mathbf{z}(t), t, \theta) \tag{2.13}$$

which is of the same form as the ordinary differential equation presented in Equation 2.1. That is to say, many of the most well known recurrent and residual architectures can be recast into an ordinary differential equation in the continuous limit (R. T. Q. Chen, Rubanova, et al. 2018). Using this logic, it is possible to parametrize any ordinary differential equation as a neural network and integrate it using a numerical solver given initial conditions. However, one of the challenges of passing a neural network output to an integrator is the fact that the integrator induces additional operations on the weights i.e. the stacked hidden layers become computationally and memory expensive to backpropagate through, which also

leads to the vanishing gradient problem. Neural Ordinary Differential Equations (NeuralODEs), via the adjoint method, allow us to integrate the neural network with constant memory cost (R. T. Q. Chen, Rubanova, et al. 2018). In brief, the adjoint method uses Lagrange multipliers to transform the gradient of the loss from a sequence of memory intensive chain rule evaluations to a single integral (R. T. Q. Chen, Rubanova, et al. 2018).

NeuralODE is used in numerous systems aimed at learning continuous time dynamics (R. T. Q. Chen, Rubanova, et al. 2018; Sanchez-Gonzalez, Victor Bapst, et al. 2019; R. T. Q. Chen, Amos, et al. 2021; Zhong and Leonard 2020). In some settings a one-step integration is applied. For this, a complex ODESolver is replaced in favour of a simple explicit integrator e.g. Runge-Kutta 4. There are indeed settings where a multi-step integration is used, such as in (Zhong, Dey, et al. 2020a) and in (Saemundsson et al. 2020) for which continuous depth networks cause severe memory issues if NeuralODE is not used. Despite their success, the work in (Dupont et al. 2019) identifies the instability of using the adjoint method to compute continuous depth networks and proposes a method to resolve this issue.

Time-series forecasting with neural networks is therefore deeply reliant on NeuralODE-like systems that are capable of replicating the behaviour of dynamical systems. For many well known physical systems, using NeuralODE is a natural temporal bias as it approximates the derivatives of a differential equation with data.

### 2.2.4 Physics Informed Neural Networks

Physics informed neural networks (PINNs) (Maziar Raissi et al. 2017), are neural networks designed to solve partial differential equations. Formally, PINNs can be trained to solve a general differential equation of the form:

$$\mathcal{N}[u] = f \tag{2.14}$$

where $u$ is the solution, $f$ is a non-homogeneous force and $\mathcal{N}$ is a linear or non-linear differential operator that extracts partial derivatives of the solution. To solve this equation, given initial and boundary conditions of $u$ and collocation points (points

of evaluation), PINNs first compute the approximate solution, $u$, with a neural network. Then, using backpropagation, PINNs compute partial derivatives of the output with respect to the input variables $\frac{\partial^n u}{\partial i^n}$ where $i$ denotes the variable and $n$ denotes the order of the derivative. An L2 loss is then used to minimize the predicted $\hat{u}_{\text{ic}}$ vs. the ground truth $u_{\text{ic}}$ (for initial and boundary conditions) coupled with a penalization of the differential equation at specific collocation points. The loss is:

$$\mathcal{L} = \mathcal{L}_{\text{diffeq}} + \mathcal{L}_{\text{conditions}} \tag{2.15}$$

$$\mathcal{L} = |\mathcal{N}[\hat{u}] - f|^2 + |u_{\text{ic}} - \hat{u}_{\text{ic}}|^2. \tag{2.16}$$

By enforcing the two loss functions, the solution $u$ can be approximated using a network. This approach has been extensively adopted across many applications including non-linear structures (R. Zhang et al. 2020), fluid flow on large domains (H. Wang et al. 2021), moving boundaries (S. Wang and Perdikaris 2021), inferring micro bubble dynamics (Zhai et al. 2021a), cardiac activation mapping (Sahli Costabal et al. 2020), ocean modelling (Wolff et al. 2021), bundle solvers (Flamant et al. 2020) and stochastic and high-dimensional PDEs (Karniadakis et al. 2021; Yang et al. 2018; Sirignano et al. 2018). One limitation to this approach is the necessity of the functional form of $\mathcal{N}$. However, to address this, the authors of PINNs also developed deep hidden physics models (M. Raissi et al. 2019) that are designed to learn the functional form without any prior knowledge of the function $\mathcal{N}$ from data. The idea behind these deep hidden physics models is to compute $u$ and then compute up to the k-th order derivatives of $u$. Using a regressive search, one can then identify the components that make up the underlying equation needed to satisfy noisy data.

PINNs are therefore a deeply powerful approach to learning differential equations that provide numerous advantages over classic numerical techniques in solving high dimensional partial differential equations. Their core operation hinges on determining partial derivatives with autograd, a technique which is now being extensively used to identify Hamiltonian dynamics. As we later show in Chapter 5, PINNs can be trained on a number of equations simultaneously and used for efficient transfer learning.

### 2.2.5 Energy Conserving Networks

Energy conserving networks aim to compute the energy of a system from data and ideally preserve it during time-series forecasting. In this section we review some of the latest advances in energy-conserving networks.

#### Hamiltonian Neural Networks

The work in (Samuel Greydanus et al. 2019) demonstrates that dynamic predictions of energy-conserving systems can be improved using Hamiltonian Neural Networks (HNNs), which endow models with a Hamiltonian constraint. The Hamiltonian, as described previously, is an important representation of a dynamical system. It relies on the Hamilton equations:

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}. \tag{2.17}$$

As a consequence, it is noted in (Samuel Greydanus et al. 2019) that by accurately learning a Hamiltonian, the system's dynamics can be naturally extracted through backpropagation, similar to (M. Raissi et al. 2019). This information thus allows us to build two 1st-order differential equations which can be used to update a predicted state space, $(\mathbf{q}, \mathbf{p})$. Equation 2.18 shows this integral, in which we define the symplectic gradient $\mathbf{S} = \left[\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}\right]$:

$$(\mathbf{q}, \mathbf{p})_{t+1} = (\mathbf{q}, \mathbf{p})_t + \int_t^{t+1} \mathbf{S}(\mathbf{q}, \mathbf{p}) \mathrm{d}t \tag{2.18}$$

Therefore, by guiding the network to learn an approximate Hamiltonian, it is possible to constrain the network to satisfy Hamilton's equations and learn energy-conserving dynamics from data that exhibits energy conservation.

#### Variational Integrator Networks

Lagrangian mechanics presents an alternative formalism to the Hamiltonian where the action integral (Equation. 2.4) and the Euler-Lagrange Equations of Equation. 2.5 can be solved instead. Typically, this is done by discretizing the equations and using variational calculus to solve the integral. It can be shown that doing

so results in variational integrators - integrators that preserve the symplecticity of the physical system and thus, the energy. The work in (Saemundsson et al. 2020) shows that, by adopting this approach, one can develop Variational Integrator Networks (VINs) which make network learning in noisy data-settings more robust. Similar to Hamiltonians, Lagrangians in classical mechanics are also connected to the kinetic energy $\mathcal{T}$ and potential energy $\mathcal{V}$ via:

$$\mathcal{L} = \mathcal{T}(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{V}(\mathbf{q}, \dot{\mathbf{q}}) \tag{2.19}$$

Furthermore, Variational Integrators are symplectic and momentum conserving (Lew et al. 2004). In other words, not only is the energy constraint enforced by the Lagrangian, but it is also enforced through the integrator. By embedding the combined inductive bias, VINs significantly outperform HNNs in learning from noisy data.

**Lagrangian Neural Networks**

Lagrangian Neural Networks (LNNs) (Cranmer, Sam Greydanus, et al. 2020) aim to tackle the problem of learning from canonical coordinates. Many datasets do not usually consist of canonical position and momentum, rather they use generalized coordinates. Furthermore, the methods in (Saemundsson et al. 2020; Lutter et al. 2019) make the assumption that the generalized $\dot{q}$ and canonical $p$ momenta are related by $p = M(q)\dot{q}$ where $M(q)$ is an inertia matrix. By reformulating the problem, LNNs do not assume any form for the connection between generalized and canonical momenta. This is achieved by vectorizing the Euler-Lagrange equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{q}}\mathcal{L} = \nabla_q \mathcal{L}, \tag{2.20}$$

Then, using the chain rule to expand the time derivative by allowing $\nabla_{\dot{q}}\mathcal{L}$ to be a function of $q$ and $\dot{q}$, they obtain:

$$(\nabla_{\dot{q}}\nabla_{\dot{q}}^T\mathcal{L})\ddot{q} + (\nabla_q \nabla_{\dot{q}}^T \mathcal{L})\dot{q} = \nabla_q \mathcal{L}. \tag{2.21}$$

Using matrix inversion, LNNs are able to obtain $\ddot{q}$ without having made any simplifying assumptions about the problem under investigation. The authors show that such an approach can be exploited to solve the chaotic double pendulum as well as the wave equation.

### 2.2.6 Energy Non-Conserving Networks

All the methods in the previous section dealt with energy-conserving priors. However, real-world systems typically experience external forcing and dissipation. To learn the dynamics of such systems, one needs to extend and generalize Hamiltonian and Lagrangian dynamics.

#### Deep Lagrangian Networks

Deep Lagrangian Networks (DeLaN) (Lutter et al. 2019) were the first networks to use a Lagrangian embedded in a neural network to learn the dynamics of a forced system. The Euler-Lagrange equations, earlier described, can be generalized to externally forced systems as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau \tag{2.22}$$

where $\tau$ represents generalized forces. DeLAN uses multiple network heads to compute individual components of the equation above. The important result here is that DeLAN can learn the dynamics of a forced system e.g. a double pendulum with a control signal that mimics the behaviour of a robotic arm.

#### Symplectic ODE Net

Symplectic-ODE Networks (Zhong, Dey, et al. 2020a) extend HNNs into the control domain. If external control is affine and influences the change in generalized momenta then the state-space derivatives can be written as:

$$\begin{bmatrix}\dot{q} \\ \dot{p}\end{bmatrix} = \begin{bmatrix}\frac{\partial H}{\partial p} \\ -\frac{\partial H}{\partial q}\end{bmatrix} + \begin{bmatrix}0 \\ g(q)\end{bmatrix} u \tag{2.23}$$

where $g(q)u$ represents a control variable. If the rank of $g(q)$ is the rank of $q$ then the system is fully actuated (Zhong, Dey, et al. 2020a). For such systems, a controller $u = \beta(q) + v(p)$ can be designed to change the potential energy landscape so as to force a system toward a specific configuration. Arguably this is the first paper to think extensively about the bottlenecks presented in HNNs. The authors extend HNN to learn from systems that experiences external control as well as resolve systems that need to be driven to a specific configuration via energy shaping.

**Unsupervised Learning of Lagrangian Dynamics**

In (Zhong and Leonard 2020), the authors show that Lagrangian dynamics can be learned from visual data. The paper introduces a co-ordinate aware variational autoencoder to encode the latent space. Using the encoded latent space, a Lagrangian is learnt so that the time derivatives of the latent space can be extracted. The paper shows that the motion of a pendulum can be learnt from visual data. More importantly, via energy shaping, the pendulum can be constrained into a specific configuration of choice. As a prototype, this paper is particularly powerful as it suggests that physics informed priors can help us learn accurate dynamics from visual data, an open challenge in machine learning.

### 2.2.7 Symplectic Networks

Many researchers have identified that embedding integrators into the learning process of dynamic systems alleviates the need for the derivatives $[\dot{q}, \dot{p}]$ of the state vector $[q, p]$. That is, with an embedded integrator such as NeuralODE, the only ground truth data needed to optimize the network is that of the state. This implies that gradient information, typically expensive to acquire, is not required. However, as (Zhu et al. 2020) and a number of other researchers highlight, the choice of integrator can play a significant role in the accuracy of long range predictions. In principle, integrators that preserve symplecticity show significantly better long range results because they enforce phase-space volume conservation (see Appendix. A.4).

We report some of the main breakthroughs in embedding symplectic integrators into networks here.

**Symplectic Recurrent Neural Network**

Symplectic RNN (Z. Chen et al. 2020) shows that for low order methods, using a variational/symplectic integrator is significantly more advantageous than using a non-symplectic Runge-Kutta method. Despite this, the paper does not discuss including higher-order integration schemes and comparing them. To tackle this, we conduct a systematic investigation of a range of integrators in Chapter. 3.

**Deep Hamiltonian Networks Based on Symplectic Integrators**

This paper (Zhu et al. 2020) reviews the integrator of choice for HNNs. One of its primary objectives is to establish the difference in performance between using a symplectic vs non-symplectic integrator from a theoretical perspective. Quite evidently, using a symplectic integrator allows us to conserve the phase-space volume of the system - a crucial component in preserving energy. Their results reiterate the need for symplectic integrators when dealing with Hamiltonian-like systems.

**SympNets**

While all the methods discussed so far exploit backpropagation to compute partial derivatives, an alternative is proposed in (Jin et al. 2020). The method avoids the need for a separable Hamiltonian and more importantly, is designed to eliminate the need for backpropagating the Hamiltonian with respect to the input. In this framework, a sequence of symplectic maps are used to transform the input into the output. The symplectic map can be split into an upper triangular matrix and lower triangular matrix with diagonals set to 1. Non-diagonal terms are parametrized by a neural network and by stacking a range of symplectic maps together, one can learn dynamics better. In fact, the results show that by doing this, the learned trajectory is significantly more accurate than HNNs with symplectic integrators. The introduction of SympNets takes us in a direction orthogonal to the concept of backpropagating a function with respect to its input.

### 2.2.8 Gaussian-Based Physics Networks

To quantify uncertainty in the models considered, Gaussian-based networks have emerged. For example, the work of (Geist et al. 2020) introduces a Gaussian Process (GP) model that utilises mechanical constraints as prior knowledge for learning system dynamics. The GP is constrained to satisfy Gauss' Principle. Using the Udwadia-Kalaba equation, the acceleration of a particle can be disentangled into an unconstrained acceleration, an ideal constraint and a non-ideal constraint. Using this knowledge, one can feed this structural form into the mean of a Gaussian Process regression. The process can be used to set priors on unknown parameters linked to the constraints. The proposal is important because it tackles constraint optimisation of physics based networks using GPs which can give us explicit uncertainty bounds on our learned trajectories.

Furthermore, in (Atkinson 2020), the author presents a novel approach to tackle learning from noisy data. The paper is arguably the first method to combine Gaussian Processes with PINNs in a unified manner. A Gaussian process prior is placed over a range of differential operators. By optimizing a network, the GP is able to find the best fit differential operators given data. The paper is thus able to identify the underlying differential equation from data but leverages PINNs and Gaussian Processes to do so. The main breakthrough of using this approach is uncertainty quantification of the learned operators and functions that describe the differential equation as well as the ability to inspect convergence.

Needless to say, uncertainty quantification in this domain is gaining significant attention and being embedded into the learning process.

### 2.2.9 Data Driven Model Discovery

Although physics-informed priors are an emergent area within machine learning capable of strongly constraining networks, more data heavy machine learning methods have been extensively studied to identify the governing dynamics of physical systems with little to no constraints.

### Discovering Physical Concepts with Neural Networks

Although this review has looked closely at embedding physical laws into neural networks, some approaches such as (Iten et al. 2020) take on an approach with very weak constraints. By feeding inputs into an encoder to learn a representation and then querying this representation before decoding it, the experimental process of learning laws is embedded in the learnt representation. The only caveat to this modeling process is the use of disentangled variational autoencoders that are designed to produce orthogonal representations. The assumption here being that every new variable added to the representation should be independent and alter the output landscape separately i.e. the variables should span the output space. Results of such an approach illustrate that the network can learn accurate latent representations necessary to evolve the time dynamics of a system, including conservation laws. This is a particularly powerful outcome as it indicates how networks are able to learn accurate representations needed to forecast dynamical systems. However, it should be noted that such systems typically require much more data and their learnt representations suffer from a lack of interpretability.

### Symbolic Pregression

Symbolic pregression builds an unsupervised framework for discovery of differential equations from visual data (Udrescu et al. 2021). In their work, they illustrate that visual dynamics can be accurately learnt by discovering the underlying equations of motion in the latent space of an encoder. However, the authors do note that such systems are not perfectly identifiable and there exists a need to constrain the learnt representations to ensure convergence.

## 2.2.10 Discussion

We have seen that learning biases in neural networks significantly improve predictive performance at a range of tasks. As many of these developments are in the prototyping phase, these improvements can be categorized by their robustness. In most of the existing literature, robustness is measured by verifying that (a), models

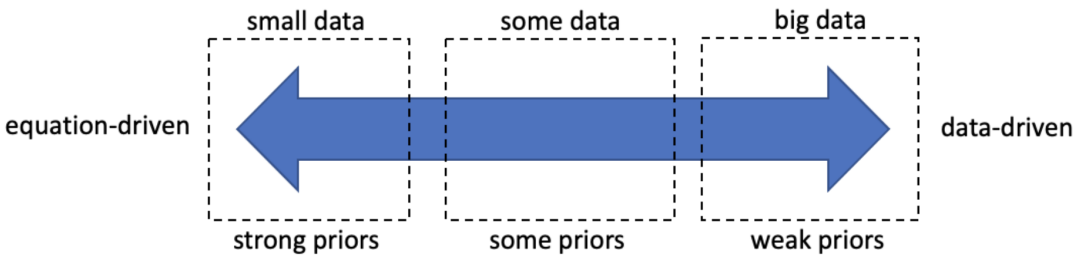



**Figure 2.2:** Spectrum highlighting the connection between priors needed and amount of data

improve performance across multiple use-cases e.g. different systems of investigation (b) testing error (measured by mean-squared error across multiple initial conditions) is consistently lower than alternative approaches and (c) the models continue to work even with noisy input data. Indeed for safety critical applications more work needs to be done to provision additional statistical guarantees for using such methods.

Importantly, these learning biases are typically categorized on a spectrum of strong to weak (Karniadakis et al. 2021). In Fig.2.2 we show the spectrum over which existing methods lie. On the far left we have equation-driven approaches that depend heavily on strong priors, such as a governing equation and that require very little data, such as initial or boundary conditions. Prime examples of such methods include solving known differential equations for specified conditions using Runge-Kutta methods or Finite Element approaches. On the far right we have fully data-driven approaches that depend heavily on large amounts of data such as images, text and state measurements and for which very little is known about the system at hand, perhaps due to its complexity. Examples of this include modelling complex turbulent flow and predicting material properties. The focus of this thesis lies at the intersection of these two ends, namely, the setting where some data as well as some priors are available. This intersection is an emergent area in scientific machine learning, attempting to bridge the divide not just across methods but also across industries. Methods such as Hamiltonian neural networks and physics-informed neural networks fall in this middle domain and demonstrate promising new directions in which both data-driven approaches can be strengthened with

priors and equation-driven approaches can be augmented with data to improve their accuracy and reduce misspecification.

In the following chapters we investigate systems that fall into the middle, or strong prior, categories as these are emergent areas in scientific machine learning.

# 3

# Variational Integrator Graph Networks for Learning Energy Conserving Dynamical Systems

# Abstract

Recent advances show that neural networks embedded with physics-informed priors significantly outperform vanilla neural networks in learning and predicting the long term dynamics of complex physical systems from noisy data. Despite this success, there has only been a limited study on how to optimally combine physics priors to improve predictive performance. To tackle this problem we unpack and generalize recent innovations into individual inductive bias segments. As such, we are able to systematically investigate all possible combinations of inductive biases of which existing methods are a natural subset. Using this framework we introduce *Variational Integrator Graph Networks* - a novel method that unifies the strengths of existing approaches by combining an energy constraint, high-order symplectic variational integrators, and graph neural networks. We demonstrate, across an extensive ablation, that the proposed unifying framework outperforms existing methods, for data-efficient learning and in predictive accuracy, across both single and many-body problems studied in recent literature. We empirically show that the improvements arise because high-order variational integrators combined with an energy constraint induce coupled learning of generalized position and momentum updates which can be formalized via the Partitioned Runge-Kutta method.

## 3.1 Introduction

Accurately and efficiently learning the time evolution of energy conserving dynamical systems from limited, noisy data is a crucial challenge in numerous domains including robotics (Lutter et al. 2019), spatiotemporal dynamical systems (Barmparis et al. 2020), interacting particle systems (Z. Li et al. 2021), and materials (Zhai et al. 2021b). To address this challenge, researchers have shown that enriching neural networks with well-chosen inductive biases such as Hamiltonians (Samuel Greydanus et al. 2019), integrators (R. T. Q. Chen, Rubanova, et al. 2018; Saemundsson et al. 2020; Chang et al. 2017) and graphs (Sanchez-Gonzalez, Godwin, et al. 2020; Battaglia, Hamrick, et al. 2018; Sanchez-Gonzalez, Heess, et al. 2018) can significantly improve the learning of complex dynamical systems over vanilla neural networks. Fundamentally, physics-informed learning biases constrain neural networks to uncover and preserve the underlying physical process of a system under investigation. Newer methods in this space now combine multiple individual inductive biases to improve overall predictive performance. However, no study extensively quantifies the performance uplift induced by any individual bias within these combinations. In addition, it remains an open challenge to identify the optimal combination for different applications.

Here, we investigate and unpack recent innovations by grouping their inductive biases into generalized segments. We then systematically investigate all possible combinations of these biases. In doing so, existing methods are naturally explored and generalized as they form a subset of the entire ablation. Using this we identify and develop *Variational Integrator Graph Networks* (VIGNs) - a novel method that brings together the core benefits of multiple inductive biases and unifies existing approaches which bring integrative, symplectic and structural form to modeling energy conserving physical systems. We also show that high-order explicit symplectic variational integrators, formalized via the Partioned Runge-Kutta (PRK) method, can be used to couple position and momentum updates for more precise learning. To benchmark our method we conduct an extensive ablation study across recent developments in physics-informed learning biases and show that

VIGNs consistently outperform existing baselines including Hamiltonian ODE Graph Networks (HOGNs) (Sanchez-Gonzalez, Victor Bapst, et al. 2019), ODE Graph Networks (OGNs) (Sanchez-Gonzalez, Victor Bapst, et al. 2019), Hamiltonian Neural Networks (HNNs) (Samuel Greydanus et al. 2019), and Variational Integrator Networks (VINs) (Saemundsson et al. 2020) across energy conserved noisy many-body dynamical systems. The outcomes of this study are therefore particularly useful for practitioners who have access to noisy position and momenta data of an energy conserving many-body system for which the underlying equations of motion are unknown and they want to: a) accurately learn the underlying dynamics such that they can predict trajectories for unseen initial conditions at inference time and, b) learn a robust network that can maintain accurate predictions beyond the training time horizon i.e. extrapolate beyond the training data range with formal guarantees that there is a valid dynamical system bridge between the last observed data and future predictions.

In section 3.2 we describe the individual learning biases that comprise VIGNs. We then outline the details of the proposed architecture in section 3.3. In Section 3.4 we demonstrate the performance of VIGNs across numerous well known energy conserving physical systems such as the simple pendulum and the many-body interacting spring particle system. Finally, in section 3.5 we unpack the performance uplift and highlight some of the limitations of the existing method.

## 3.2 Background

Numerous recent approaches tackle learning from physical data, but of them three methods stand out; Graph Networks (Sanchez-Gonzalez, Heess, et al. 2018), Hamiltonian Neural Networks (Samuel Greydanus et al. 2019) and networks with embedded integrators (R. T. Q. Chen, Rubanova, et al. 2018; Saemundsson et al. 2020). VIGNs allow us to combine the major strengths of each approach and hence form a simple, unifying framework for learning the temporal behaviour of dynamical systems. We briefly review the methods in the following sections.

| Type | MLP-Based Method | Graph-Based Method |
|---|---|---|
| Delta | **DN** $[q_1, p_1, ..., q_n, p_n] \rightarrow \text{MLP} \rightarrow [\dot{q}_1, \dot{p}_1, ..., \dot{q}_n, \dot{p}_n]$ | **DGN** $\begin{bmatrix} q_1, p_1 \\ .. \\ q_n, p_n \end{bmatrix} \rightarrow \text{GN} \rightarrow \begin{bmatrix} \dot{q}_1, \dot{p}_1 \\ .. \\ \dot{q}_n, \dot{p}_n \end{bmatrix}$ |
| Hamiltonian | **HNN** $[q_1, p_1, ..., q_n, p_n] \rightarrow \text{MLP} \rightarrow H(q,p) \xrightarrow{\text{autograd}} [\frac{\partial H}{\partial p_1}, \frac{-\partial H}{\partial q_1}, ..., \frac{\partial H}{\partial p_n}, \frac{-\partial H}{\partial q_n}]$ | **HOGN** $\begin{bmatrix} q_1, p_1 \\ .. \\ q_n, p_n \end{bmatrix} \rightarrow \text{GN} \rightarrow H(q,p) \xrightarrow{\text{autograd}} \begin{bmatrix} \frac{\partial H}{\partial p_1}, \frac{-\partial H}{\partial q_1} \\ .. \\ \frac{\partial H}{\partial p_n}, \frac{-\partial H}{\partial q_n} \end{bmatrix}$ |
| Potential | **PNN** $[q_1, ..., q_n] \rightarrow \text{MLP} \rightarrow E_{\text{potential}}(q) \xrightarrow{\text{autograd}} [\frac{-\partial E_{pot}}{\partial q_1}, ..., \frac{-\partial E_{pot}}{\partial q_n}]$ | **PGN** $\begin{bmatrix} q_1 \\ .. \\ q_n \end{bmatrix} \rightarrow \text{GN} \rightarrow E_{\text{potential}}(q) \xrightarrow{\text{autograd}} \begin{bmatrix} \frac{-\partial E_{\text{pot}}}{\partial q_1} \\ .. \\ \frac{-\partial E_{\text{pot}}}{\partial q_n} \end{bmatrix}$ |

**Table 3.1:** Three major methods used to approximate state derivatives $\dot{s} = [\dot{q}, \dot{p}]$. Delta: approximate state derivatives directly. Hamiltonian: approximate state derivatives by learning a Hamiltonian then differentiating this w.r.t. input. Potential: approximate $q$ via potential assuming that $\dot{q} = M^{-1}p$. Abbreviations (in bold) refer to the methods presented in this study.

As discussed earlier, dynamical systems can be represented by systems of ordinary differential equations of the form:

$$\dot{\mathbf{s}} = f(\mathbf{s}, t), \tag{3.1}$$

where $\mathbf{s} = (\mathbf{q}, \mathbf{p})^T$ is a state vector composed of position and momentum, $t$ is time, and $f$ is an arbitrary function of time and the state vector. Physics-informed networks attempt to approximate $f$ using a neural network. In what follows we describe three main approaches used to approximate $f$, delta networks, Hamiltonian networks, and potential networks. Network designs for the three methods are summarized in table 3.1.

### 3.2.1 Delta Networks

A straightforward approach to learning dynamics from data is to use a neural network to transform a current state vector $\mathbf{s}_t$ to a future state vector $\mathbf{s}_{t+1}$. Early research (Pukrittayakamee et al. 2009) however found that using a neural network to learn the gradients $\dot{\mathbf{s}}$ and then applying an integration results in better performance. Here, we refer to networks that directly compute $\dot{\mathbf{s}}_t$ from $\mathbf{s}_t$ as delta networks (DNs). As an aside, these networks are referred to as 'baseline' in (Samuel Greydanus et al. 2019). In (Sanchez-Gonzalez, Victor Bapst, et al. 2019) these gradients are computed with graphs and referred to as ODE Graph Networks (OGN). To

maintain consistency we refer to multi-layer perceptron (MLP) based methods as Delta Networks (DNs) and graph based methods as Delta Graph Networks (DGNs).

### 3.2.2 Hamiltonian Neural Networks

In designing a neural network for time-series forecasting, the typical operation of interest for many physical systems is one that accurately models the time evolution of the system. Recently, the work of (Samuel Greydanus et al. 2019) demonstrated that predictions through time can be improved using Hamiltonian Neural Networks (HNNs) which endow models with a Hamiltonian constraint. Given a system with $N$ particles, the Hamiltonian $\mathcal{H}$ is a scalar function of canonical position $\mathbf{q} = (q_1, q_2, ...., q_N)$ and momentum $\mathbf{p} = (p_1, p_2, ...., p_N)$. In representing physical systems with a Hamiltonian, one can simply use Hamilton's equations to extract the time derivatives of the inputs by differentiating the Hamiltonian with respect to its variables as:

$$\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}, \tag{3.2}$$

where $\dot{a} = \frac{da}{dt}\ \forall a(t)$. As a consequence, it is noted in (Samuel Greydanus et al. 2019) that by training a network to learn a scalar output $\mathcal{H}$ given inputs $[\mathbf{q}, \mathbf{p}]$, the system's state-time derivatives can be naturally extracted through auto-differentiation of the predicted Hamiltonian with respect to the inputs. In other words, the backpropagation technique traditionally used to compute gradients of the loss function with respect to the weights can also be used to compute partial derivatives of the Hamiltonian with respect to the input. The Hamiltonian in most systems represents the total mechanical energy of the system and is therefore a powerful inductive bias that can be utilized to evolve a physical state while maintaining energy conservation. Indeed such approaches can be extended to damped and forced systems via port-Hamiltonians (S. A. Desai, Mattheakis, Sondak, et al. 2021) as we later show. In addition, HNNs combined with graph networks and embedded integrators are referred to as Hamiltonian ODE Graph Networks (HOGNs) (Sanchez-Gonzalez, Victor Bapst, et al. 2019).

### 3.2.3 Potential Neural Networks

Separable Hamiltonians found in many dynamical systems can be written as $\mathcal{H}(\mathbf{q}, \mathbf{p}) = E_{\text{kinetic}}(\mathbf{p}) + E_{\text{potential}}(\mathbf{q})$. Typically, for rigid body systems, the form of the kinetic energy is $E_{\text{kinetic}} = \frac{M^{-1}}{2}\mathbf{p}^2$ where $M$ is an inertial mass matrix that connects the generalized momenta $\dot{\mathbf{q}}$ to the canonical momenta $\mathbf{p}$ such that $\dot{\mathbf{q}} = \frac{\partial E_{\text{kinetic}}}{\partial \mathbf{p}} = M^{-1}\mathbf{p}$. The authors of (Yu et al. 2020) and (Saemundsson et al. 2020) exploit this simplification when dealing with generalized position $\mathbf{q}$ and velocity $\dot{\mathbf{q}}$ to collapse Eq. 3.2 into:

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \dot{\mathbf{q}}, \quad \frac{\mathrm{d}\dot{\mathbf{q}}}{\mathrm{d}t} = -M^{-1}\frac{\partial E_{potential}(\mathbf{q})}{\partial \mathbf{q}}. \tag{3.3}$$

Equation 3.3 allows us to learn a single function $E_{\text{potential}}$ with fewer network weights needed to learn a Hamiltonian and a single backpropagation with respect to $\mathbf{q}$ as opposed to $[\mathbf{q}, \mathbf{p}]$ for HNN. Further, it gives us the flexibility to learn the inertial mass matrix by explicitly learning $M_\theta$ rather than nesting it in $\tilde{\mathcal{U}}_\theta = M^{-1}\mathcal{U}$. As this type of network has not been introduced formally as an individual inductive bias, we coin the term Potential Neural Networks (PNNs) in reference to them and Potential Graph Networks (PGNs) in reference to their graph based variants.

In the case where we only have data with canonical coordinates, potential networks can still be used but a separate neural network needs to be designed to learn the inertial matrix $M$ (Saemundsson et al. 2020).

### 3.2.4 Embedded Integrators

One approach to solving Eq. 3.1 is to parametrize the function $f$ by one of the methods just described and minimize the euclidean distance between the predicted state time derivatives $\hat{\dot{\mathbf{s}}}$ and the ground truth $\dot{\mathbf{s}}_{\text{gt}}$ derivatives. One challenge in doing this is it assumes we have access to the ground truth state time derivatives, which are typically hard to extract. To circumvent this problem, researchers embed a numerical integrator into the training process (R. T. Q. Chen, Rubanova, et al. 2018; Zhong, Dey, et al. 2020a). Formally this equates to integrating both sides of Eq. 3.1 such that:

Short Range Integration:

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \int_t^{t+\Delta t} f_\theta(\mathbf{s}, t)\mathrm{d}t. \tag{3.4}$$

Long Range Integration:

$$\mathbf{s}_{t+n} = \mathbf{s}_t + \int_t^{T_{\max}} f_\theta(\mathbf{s}, t)\mathrm{d}t, \tag{3.5}$$

where $\theta$ are the weights of the neural network. The short range integration involves a single discrete step $\Delta t$ whereas the long range integration involves a sequence of discrete steps to the final time $T_{\max} = n\Delta t$. It can be shown that if we integrate the system from $t$ to $T_{\max}$, the network above ends up being a composition of multiple transformations as would be found in recurrent neural networks and residual networks (R. T. Q. Chen, Rubanova, et al. 2018).

**Symplecticity**

While the embedded integrator resolves the challenge of having to obtain state time derivatives, it introduces a new complexity - the choice of integrator. Numerical integrators (see Appendix.A) are chosen based on their precision and truncation error, however, when dealing with Hamiltonian systems an additional factor to consider is whether the integrator preserves the energy of the system. It has been shown that symplectic integrators can preserve a symplectic form (energy) during integration making them versatile for long-range integrations of conserved physical systems (Marsden et al. 2001). A symplectic map is a smooth manifold that generalises the phase space of classical systems by defining vector fields along which total energy can be conserved. That is to say, symplectic maps preserve the energy of a classical system as they are traversed. A symplectic integrator is an integrator whose solution resides on a symplectic manifold. As such, the integrator not only has guarantees on its global and local truncation error related to the states, but also has guarantees on the truncation error of energy. Typically, if the method is p'th order accurate, the local truncation error for energy is $(h^{p-1})$. The performance of these numerical integrators in solving a range of different systems is outlined

in Appendix. A.5, where we see that for long-range integrations even low order symplectic integrators are as performant as high order Runge-Kutta (RK) methods in terms of energy conservation. While Variational Integrator Networks (VINs) (Saemundsson et al. 2020) and Symplectic Recurrent Neural Networks (Z. Chen et al. 2020) both illustrate how an embedded symplectic integrator improves learning over traditional RK methods, they only do so for low-order methods. To extend our investigation to high-order symplectic integrators, we later investigate Partitioned Runge-Kutta (PRK) methods.

Typically, RK methods and their coefficients can be described by Butcher tables (see Appendix A.2). If the coefficients satisfy certain conditions then they can be made symplectic (Marsden et al. 2001). However, the additional symplecticity constraint on the table of coefficients forces the integration scheme to be implicit. While implicit integrators are powerful, they require a root finding approach. Introducing such complexity into an embedded NN is possible but significantly complicates the backpropagation technique. As such, it is of importance to establish whether explicit symplectic methods can be developed. Fortunately, by creating separate Butcher tables for position and momentum it is indeed possible to describe a Partitioned Runge-Kutta (PRK) method with coefficients that result in an explicit symplectic integration scheme. Note that variational integrators, derived through variational calculus, can be described by PRK methods (Marsden et al. 2001). As a consequence, it is possible to generalize the result of Variational Integrator Networks (VINs) to higher order methods (see Appendix A.4 for details).

### 3.2.5 Graph Neural Networks

Until this point, we have strictly been speaking of neural networks as multilayer perceptrons (MLPs). However, interacting particle systems have structure and therefore the state of a physical system can be represented by a graph $G = (u, V, E)$ (Battaglia, Hamrick, et al. 2018). For example, a node ($V$) can be a particle in a many-body problem. These nodes can be used to represent the core features of the particle like its position, momentum, mass, and other particle constants. Edges

($E$) can represent forces between the particles, and the 'Globals' ($u$) can represent universal constants such as air density, the gravitational constant etc. In representing physical systems this way, we are able to preserve the structure of our data and find solutions that conform to this prior structure using graph neural networks(Sanchez-Gonzalez, Godwin, et al. 2020; Battaglia, Hamrick, et al. 2018; Battaglia, Pascanu, et al. 2016; Sanchez-Gonzalez, Heess, et al. 2018; Seo and Liu 2019; Cranmer, Sanchez Gonzalez, et al. 2020; Seo, Meng, et al. 2020; Lamb et al. 2020; Cranmer, Sam Greydanus, et al. 2020). Graph neural networks carry out a sequence of transformations to the graph nodes and edges to update the graph parameters. The representation is therefore powerful for many-body systems primarily because the graph networks can operate within the known constraints of physical systems.

### 3.2.6 Related work

The notion of embedding physically-informed inductive biases in neural networks can be found in numerous early work aimed at modeling materials (Rupp et al. 2012; Smith et al. 2017; Witkoskie et al. 2005; Pukrittayakamee et al. 2009; Yao et al. 2018). For example, early efforts (Witkoskie et al. 2005) demonstrate that in contrast to directly learning a potential energy surface, the inclusion of gradients in the learning process can drive a network to accurately model the forces. However, most materials modeling frameworks are task-specific and usually do not generalize well.

More general approaches that capture physical laws include search algorithms (Hills et al. 2015), symbolic learning (Cranmer, Sanchez Gonzalez, et al. 2020), as well as regressive techniques (Iten et al. 2020; Schmidt et al. 2009; Silva et al. 2020). In addition, graphs have also been presented as natural inductive biases in modeling physics (Battaglia, Hamrick, et al. 2018; Battaglia, Pascanu, et al. 2016).

NeuralODE (R. T. Q. Chen, Rubanova, et al. 2018) has also re-sparked an interest in inductive biases for differential equations. Inspired by this work, (Samuel Greydanus et al. 2019) and (Toth et al. 2020) show that a neural network can be used to predict a Hamiltonian which can be differentiated with respect to the input ($\mathbf{p}$ and $\mathbf{q}$) to obtain the time derivatives of the system. With these derivatives accurately

learnt, a NeuralODE-type integration scheme can be used to evolve a system. This general approach has formed the basis for many advancements within physical learning (Sanchez-Gonzalez, Godwin, et al. 2020; Sanchez-Gonzalez, Victor Bapst, et al. 2019; Zhong, Dey, et al. 2020a; Saemundsson et al. 2020; Choudhary et al. 2020).

Although HNNs predict dynamics for few body systems well (e.g. a swinging pendulum or mass spring system) they are not readily adaptable to large N-body problems when the input dimension grows. The work in (Sanchez-Gonzalez, Victor Bapst, et al. 2019) shows that graph networks are ideal for resolving this type of system because they can operate on structured data i.e. the system does not need to be vectorized as would be the case for multi-layer feed forward neural networks.

Inspired by NeuralODEs, variational integrator networks (VINs) (Saemundsson et al. 2020) propose a neural network whose architecture matches the discrete equation of motion governing the dynamical system, as derived by applying the Euler-Lagrange equations to a discretized action integral. The paper indicates major benefits when using the method for noisy data, as well as providing precise energy and momentum conservation.

While it is clear that the constrained HNN (Finzi, K. A. Wang, et al. 2020) is capable of solving Hamiltonian systems more efficiently, it assumes we have access to Cartesian coordinates for all systems and requires explicit rigid body constraints.

Our method brings together the inductive biases presented in all these papers and leverages them to solve large many-body problems in noisy data settings.

## 3.3 Method

The architecture for our method is shown in Fig. 3.1. At training time it is assumed that, for any given system, we have access to noisy state variables $\mathbf{s}(t) = [\mathbf{q}(t), \mathbf{p}(t)]$ for every particle and we know that the total energy of the system is conserved. The unknowns are the Hamiltonian, the underlying differential equations, and the noise. The objective for all the networks is to therefore approximate the underlying equations of motion. Unlike delta networks or Hamiltonian networks, our network takes as input state vectors $\mathbf{s} = (\mathbf{q})$ representing generalized coordinates and

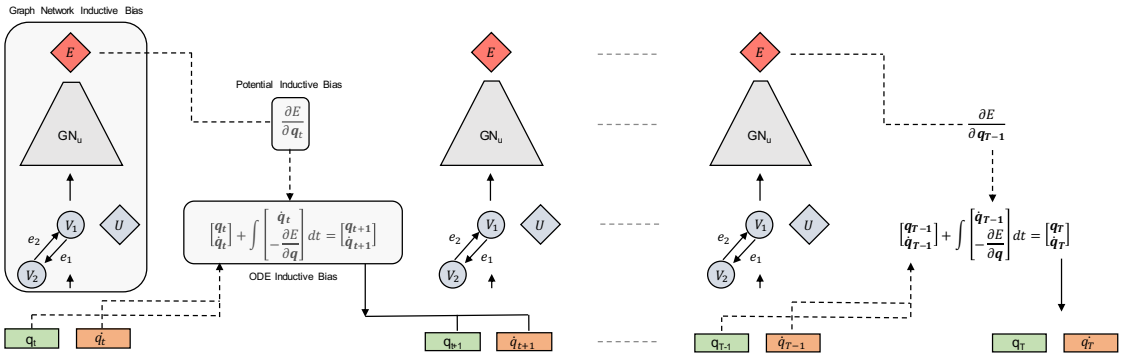



**Figure 3.1:** The architecture for our method takes as input the position vector $[q]$ and feeds it through a graph network to compute the potential energy $E(q)$. Using backpropagation, the update for $\dot{q}$ is computed and the input state is integrated one step. Continuing this sequence yields an multi-step integration scheme.

learns to predict the potential function and its derivatives with respect to the inputs (potential networks). Note, we adopt the potential neural network so we require generalized coordinates. However, the transition to a Hamiltonian NN is straightforward. Since the input training data can be described by a graph, we show vertices $V_i$, edges $E_{ij}$ and globals $u$ as input to the graph network $GN_u$ to predict the potential energy $E_{\text{potential}}$. The graph nodes represent particle positions and momenta. Edges represent connectivity such as spring coefficients and globals contain global variables such as gravitational constants. Note that the graphs used are fully connected but this does not have to be true, as sparseness can be enforced through the graph structure as would be the case in complex particle interactions for which near-field effects are orders of magnitude stronger than far-field interactions. In addition, it is assumed that the graph structure is known a priori. For all settings, unitary edge weights are used since edge attributes in our simulated data are typically set to 1 or to constants without loss of generality. The key difference between our graph approach and HOGN (Sanchez-Gonzalez, Victor Bapst, et al. 2019) is that our network only takes the position **q** as input i.e. the nodes of the graph only have position.

Succinctly, for delta networks and Hamiltonian networks the input to the network is a state vector into the MLP or Graph Network of the form $[q, p]$ i.e. both position and momenta are fed in to learn $[\dot{q}, \dot{p}]$. For potential networks an assumption is

made about the form of $\dot{q} = M^{-1}p$ and, therefore, the only input required is the position since the only thing that needs to be learnt is $\ddot{q}$.

In the noiseless setting, for all systems the training loss is defined as the mean-squared error (MSE) across all time steps and across all state vectors such that:

$$\mathcal{L} = \frac{1}{N} \sum_{t=1}^{T_{max}} (\mathbf{s}(t) - \hat{\mathbf{s}}(t))^2 \tag{3.6}$$

where $\mathbf{s} = [\mathbf{q}, \mathbf{p}]$. Note that this is a standard approach in data-driven discovery of Hamiltonian/Lagrangian systems.

In the noisy setting, we follow a similar approach to (Saemundsson et al. 2020) and compute the full log-likelihood of the predicted state vector $\mathbf{s}_{\text{pred}}$ as:

$$P(\mathbf{s}_{\text{pred}}|\mathbf{s}, \sigma^2) = \prod_{t=1}^{T_{\text{max}}} \mathcal{N}(\mathbf{s}_{\text{pred}}(t)|\mathbf{s}(t), \sigma^2 I), \tag{3.7}$$

where $\mathcal{N}$ is a Gaussian distribution, $\sigma^2$ reflects the variance and $I$ is an identity matrix.

To benchmark the performance of our method we conduct an extensive ablation study. Our ablation iterates across all combinations of the inductive biases described in the section 3.2. Namely, it includes both graph and non-graph methods that either learn the state derivatives directly (Sanchez-Gonzalez, Victor Bapst, et al. 2019; Samuel Greydanus et al. 2019), the Hamiltonian (Hamiltonian networks) or the potential function (potential networks). We use 1st through 4th order embedded integrators that are either symplectic or non-symplectic. We also iterate over a multi-step integration scheme with step sizes of 1, 5 and 10 to account for both short, mid and long-range integrations during training. Note that we can indeed integrate for more than 10 steps but this increases the memory requirement. In addition, since the ablation iterates over all possible combinations of inductive biases, existing methods in the literature are naturally covered. For example, VINs can be described as low-order, long-range symplectic integrators coupled with potential networks. While HOGNs couple low and high-order, short-range integrators with Hamiltonians and graphs. These connections are summarized in Fig. 3.2. Since our work iterates across all these methods we adopt a new naming convention for

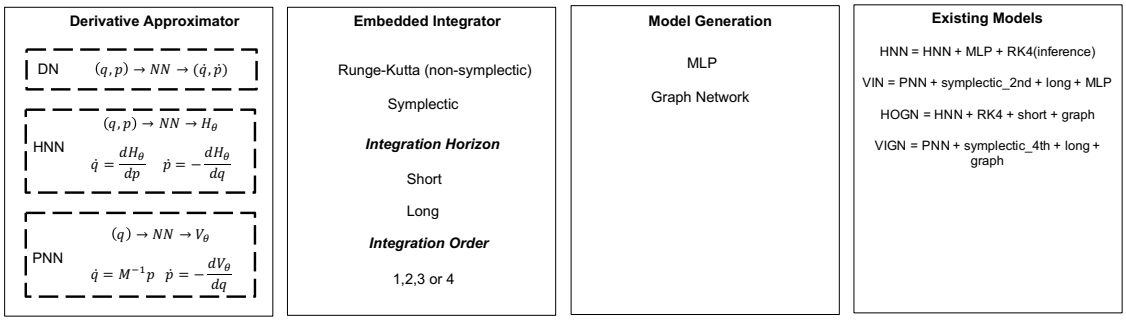



**Figure 3.2:** We define some new terms in our study that are relevant to generalizing and systematically investigating inductive biases. As such, we highlight the core independent inductive biases in the figure, as well as how existing methods are a natural combination of these components.

convenience. We refer to networks that combine graphs with potential networks as Potential Graph Networks (PGNs). For reference note that variational integrator graph networks (VIGNs) are PGNs under symplectic integration.

## 3.4 Experiments

We carry out our experiments on numerous datasets used in recent literature and describe their underlying equations herein. The train/test regime, network nodes and parameters swept over are described in Tables 3.2 and 3.3. All code is implemented in Tensorflow with Sonnet and can be found here[1]. For the non-graph based methods we use standard MLPs, where the number of hidden nodes and layers can be found in Table 3.2. The optimizer is Adam (Kingma et al. 2015) and the learning rate is $10^{-3}$ for all systems. For the graph-based methods we use Deepminds graphnet [2] package. For the graph-based methods we set the edges to be constant as we pass the spring constants to the nodes. We use both softplus and tanh activations in the ablation and report the best score of the two. The batch size of 200 is used for all systems to maintain consistency and fairness while reaching the GPU limit for the graph-based methods. For most of the single-body systems we use 10000 training iterations. For larger systems, we want to guarantee

[1] https://github.com/shaandesai1/VIGN
[2] https://github.com/deepmind/graph_nets

the best convergence so we train for 20000 iterations. To ensure robust training, we randomize the training batches.

**Table 3.2:** Training and Testing Parameters

| | Tmax Train | Tmax Test | $\Delta t$ | Samples | Training Points | Total Samples | Hidden Layers | Nodes per Layer |
|---|---|---|---|---|---|---|---|---|
| Mass Spring | 3 | 9 | 0.1 | 30 | 25 | 750 | 2 | 200 |
| Pendulum | 3 | 9 | 0.1 | 30 | 25 | 750 | 2 | 200 |
| 2-Body Gravitational | 20 | 60 | 0.1 | 200 | 20 | 4000 | 2 | 300 |
| 3-Body Gravitational | 2 | 3 | 0.1 | 20 | 200 | 4000 | 2 | 300 |
| 5 Spring Particle | 4 | 12 | 0.1 | 40 | 100 | 4000 | 2 | 300 |
| Heinon Heiles | 2 | 3 | 0.1 | 20 | 100 | 2000 | 2 | 300 |

**Table 3.3:** Parameter Sweep

| System | Model | Embedded Graph | Embedded Integrator | Integrator Type | Range | Noise |
|---|---|---|---|---|---|---|
| Mass Spring | Baseline | Yes | Yes | RK1 | 1-step | Yes |
| Pendulum | Hamiltonian | No | No | RK2 | 5-step | No |
| 2-Body Gravitational | Potential | | | RK3 | 10-step | |
| 3-Body Gravitational | | | | RK4 | | |
| 5 Spring Particle | | | | VI1 | | |
| Heinon Heiles | | | | VI2 | | |
| | | | | VI3 | | |
| | | | | VI4 | | |

**Training:** For all the systems we investigate the training data is generated using an 8th order Runge-Kutta method with $r_{\text{tol}} = 10^{-12}$ so that the ground truth is precise and conserves energy. The noise model for all systems is chosen to maintain a noise-to-signal ratio of less than 30% which allows us to investigate which architecture is the most robust to noisy data. For all noisy training configurations, the noise source is a Gaussian $\mathcal{N}(0, \sigma)$. The noise is added to each state vector similar to the approach taken in (Samuel Greydanus et al. 2019) and (Saemundsson et al. 2020). From our study of integrators (Appendix.A), we find that most methods are comparable for very small $\Delta t$. We therefore investigate all systems when $\Delta t = 0.1$.
**Testing:** To evaluate the performance of our models, we sample 50 initial conditions and integrate these systems to 3 times the training time horizon $3T_{\text{max}}$. In other words, the true performance of the model is tested by evaluating points beyond the training regime. For each set of initial conditions we compute the MSE across the entire trajectory between the prediction and the ground truth states. Note that some of our 50 sampled initial conditions can be slightly outside the training regime which can lead to a few poorly predicted trajectories by all the models. To prevent

this skewing our final reported results, we compute the geometric mean, a measure of central tendency, of the MSEs computed across the 50 initial conditions.

Here, we describe the systems investigated. A list of all the experimental results can be found in Appendix B. In the following discussion, we only report the results of our ablation with 4th order methods for clarity. The results for these systems are summarized in Fig. 3.3.

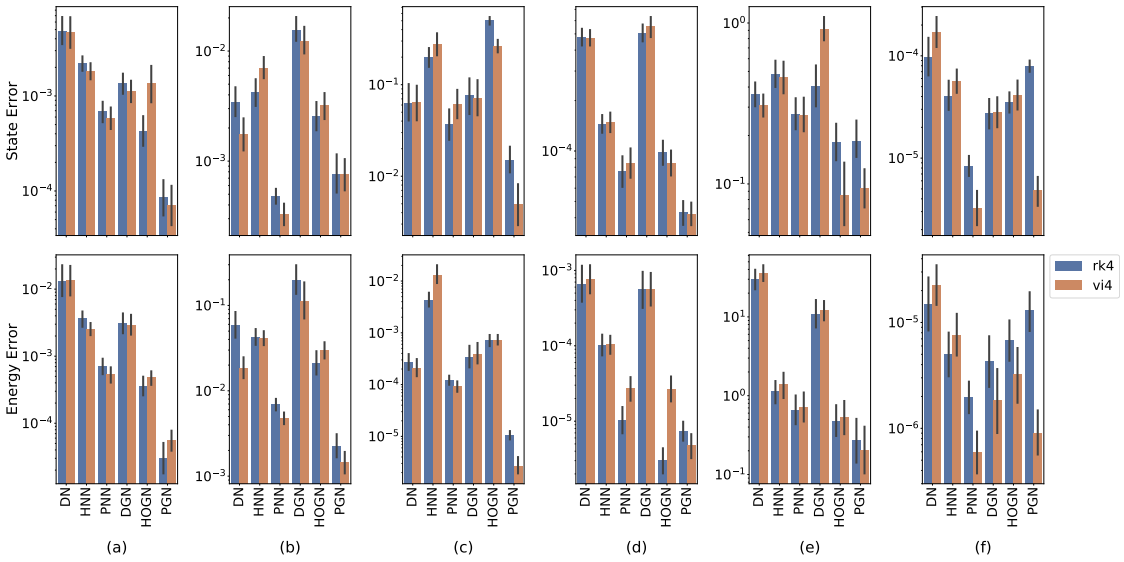



**Figure 3.3:** State and energy geometric mean (and $\pm\sigma$ standard errors) of the rollout MSE for 50 initial conditions of the (a) mass-spring, (b) pendulum, (c) 2-body gravitational, (d) 3-body gravitational, (e) 5-body spring force, and (f) Henon-Heiles systems
. The results are reported for models trained on noisy data. We see that high order, long-range integrators coupled with potential networks perform the best with their graph variants showing added versatility in both single and many-body systems.

## Mass-Spring system

We start by considering a simple frictionless 1-D mass-spring system modeled by the Hamiltonian as:

$$\mathcal{H} = \frac{p^2}{2m} + k\frac{q^2}{2}. \tag{3.8}$$

For simplicity we set the mass $m$ and spring constants to 1 without loss of generality. As is done in (Samuel Greydanus et al. 2019), the training data is sampled uniformly in an energy range of 0.5 to 4.5.

## Pendulum system

We carry out testing on a 1-D pendulum, which is more complex than the simple mass spring because it is a non-linear system. The Hamiltonian is modeled as:

$$\mathcal{H} = \frac{p^2}{2ml^2} + mgl\left(1 - \cos(q)\right), \tag{3.9}$$

where the mass and lengths are set to 1, $g$ is set to 9.81. We use 25 initial conditions which satisfy the condition that the total energy lies in $[1.3, 2.3]$ for training. Note that this energy yields strong non-linear behaviour.

## 2-body gravitational system

The 2-body system represents a particle system in which the forces between particles is modelled by a gravitational force. The system can be represented by the Hamiltonian:

$$\mathcal{H} = \sum_{i=1}^{2} \frac{|p_i|^2}{2m_i} - \sum_{1 \le i \le j \le 2} \mathrm{g} \frac{m_i m_j}{|q_j - q_i|^2}, \tag{3.10}$$

where we set masses to 1 and $g$ to 1 without loss of generality. The coordinates are assumed to be scaled by the reduced mass $\mu$, in addition, the center of mass is assumed to be fixed at 0. We use 20 initial conditions which satisfy the condition that the radius of a particle's trajectory is uniformly sampled between $[0.5, 1.5]$ as is done in (Samuel Greydanus et al. 2019). We visualize the rollout of a single test point in Fig. 3.4.

## 3-body gravitational system

The 3-body system represents a particle system in which the forces between particles is modeled by a gravitational force. The system can be represented by:

$$\mathcal{H} = \sum_{i=1}^{3} \frac{|p_i|^2}{2m_i} - \sum_{1 \le i \le j \le 3} \mathrm{g} \frac{m_i m_j}{|q_j - q_i|^2}, \tag{3.11}$$

where we set masses to 1 and $g$ to 1.

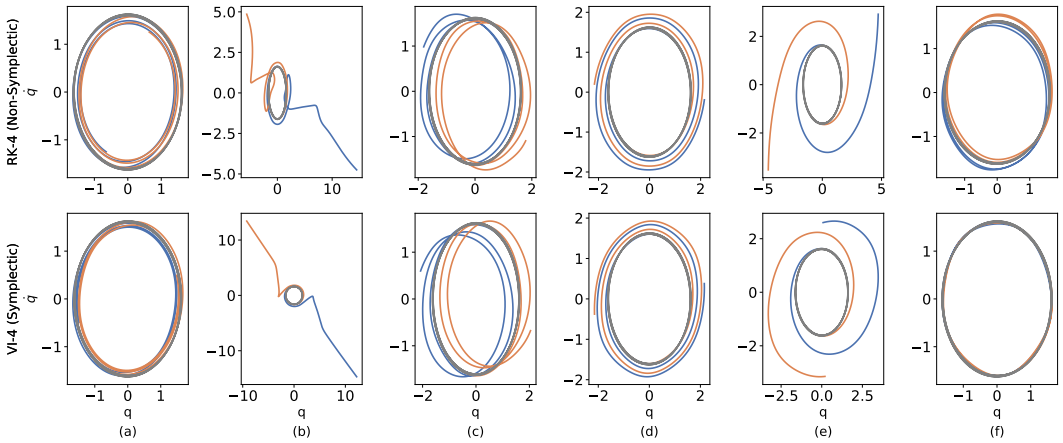



**Figure 3.4:** Qualitative evolution of a single test state of the 2-body gravitational problem trained on noisy data for (a) DNs, (b) HNNs,(c) PNNs, (d) DGNs, (e) HOGNs, and (f) PGNs. We see that PGN (f) is the most performant method as it stays close to the ground truth lines (marked in black) with the variational integrator variant of PGN (VIGN) doing the best (bottom right).

## N-body spring force system

We also carry out our experiments on a dataset similar to that found in (Sanchez-Gonzalez, Victor Bapst, et al. 2019). We develop a N-body dataset, where the interaction force between particles is modeled by $\mathbf{F}_{ij} = -k_i k_j(\mathbf{q}_i - \mathbf{q}_j)$ following the same sampling procedure in (Sanchez-Gonzalez, Victor Bapst, et al. 2019), leading to a Hamiltonian as:

$$\mathcal{H} = \frac{1}{2}\sum_{i}^{N}\frac{|\mathbf{p}_i|^2}{2m_i} + \sum_{i}^{N}\sum_{i<j}^{N}\frac{1}{2}k_i k_j(\mathbf{q}_i - \mathbf{q}_j)^2. \tag{3.12}$$

The overall mechanism closely aligns with important problems in N-particle systems used to model complex materials in solid state physics. The results of a rollout of a single point in the test set can be found in Fig.3.5.

## Hénon-Heiles system

The systems investigate so far do not exhibit chaotic motion. Hénon-Heiles is a system used to describe the nonlinear motion of a star around a galactic center and defined by the Hamiltonian:

$$\mathcal{H} = \frac{1}{2}\mathbf{p}^2 + \frac{1}{2}\mathbf{q}^2 + \lambda\left(q_x^2 q_y - \frac{q_y^3}{3}\right), \tag{3.13}$$

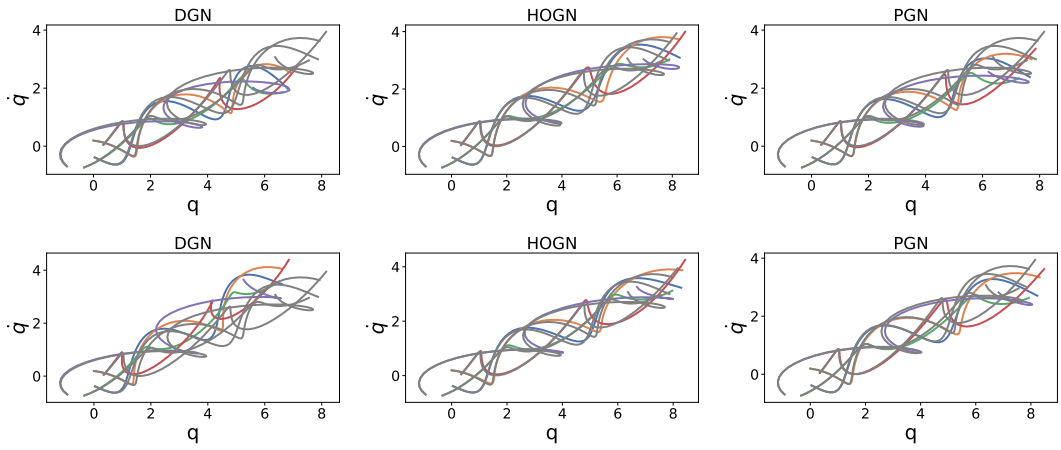



**Figure 3.5:** Rollout of a single test point of the 5 body spring particle system with each model trained on noisy data. Each color represents a different particle. The ground truth is highlighted by black lines. The top row shows each method integrated with a RK4 integrator and the bottom 4th order symplectic integration. Unlike OGN, PGNs perform very well under symplectic integration and typically outperform HOGN in energy conservation.

which exhibits chaotic motion, i.e. small perturbations on initial conditions lead to drastically different trajectories. It has been shown that Hamiltonian Neural Networks can be used to capture dynamics in this setting (Choudhary et al. 2020; Mattheakis, Sondak, et al. 2020).

We show the test results of all systems in Fig. 3.3 with models trained on noisy data using 10-step integration during training. Most notable is that potential-based networks consistently perform the best with potential graph networks doing the best in terms of state and energy MSE for most systems. In addition, the performance of RK-4 and the symplectic 4th order integrator are relatively comparable across most systems, indicating 4th order symplecticity constraints are more relevant for very large integration time steps such as in our 2-body problem or chaotic trajectories like Hénon-Heiles.

## 3.5 Conclusion

We run an extensive ablation to assess the performance of a combination of inductive biases and consistently find VIGN to be the most performant. To measure the statistical significance of our experiments, MSEs were compared to verify that one

method is better than another. We measure performance across multiple systems which we train in a specified domain and then test. Our ablation of VIGN covers over 2000 experiments, in all of which we find MSEs generated by methods with well-chosen priors to be much lower than those without. Safety is paramount for many engineering applications - as our models remain approximations, they remain susceptible to error accumulation and adversarial attack, problems that require further investigation in the future.

From our extensive ablation across both noisy and non-noisy training data we find that VIGN is consistently the best or next best method in the noisy data setting. We believe that the reason inductive biases are not as successful with noiseless data is that they overfit to the training set in addition to the networks attempting to compensate for the error induced by numerical integration. Noise naturally reduces the overfitting and thus allows VIGN/PGN to do well. Once we identified VIGN as the most performant we needed to establish which biases were most useful. From Fig. 3.3 we see that a potential network bias drives the largest performance increase against other approaches. We see that graph based methods are more performant for larger many-body systems as is expected but remain robust in the single body settings too. We also find that using a long-range integration scheme for training in the noisy data setting tends to help the overall performance of all methods as it encourages the network to learn the underlying dynamics using multiple noisy points rather than one. Indeed, we notice that performance depends heavily on the system under investigation. For example, in Fig. 3.4, we observe that using a Hamiltonian constraint can lead to diverging trajectories at inference time for the 2-body system, when noisy training data are considered. Similar performance of HNNs trained under noisy data has been reported in (Saemundsson et al. 2020), implying that networks with Hamiltonian constraints are capable of overfitting to the noise at training time and thus, occasionally, fail to identify the underlying dynamics. Hence, the choice of the prior, defining the embedded neural network physics, can play a dominant role. The choice of prior should therefore be motivated by working with

practitioners and keeping humans-in-the-loop as well as by referencing extensive ablations such as this one to identify models that work across a number of domains.

Although we do note that symplectic integrators are good for long range integration and energy preservation, their performance in many of the systems we investigate is only marginally better than Runge-Kutta. In preserving the energy, symplectic integrators are capable of drifting from the ground truth state while ensuring energy conservation which explains why we occasionally see RK methods doing better at state and energy conservation. However, we do note that symplectic integrators of low order are much better at preserving the dynamics over low order Runge Kutta methods. This result is consistent with the theory of symplectic integrators. It is important to note that the future prediction horizon depends on the accuracy of our approximation of the differential equation and the order of the embedded integrator. Assuming the approximation is good, the order of the numerical integrator determines the horizon. Importantly, the approximation depends heavily on the training data. As such, it is difficult to quantify precisely how long we can integrate, particularly for particle systems that diverge in phase space.

We have shown that learning dynamics from data strongly benefits from well-chosen inductive biases. We present VIGNs as one such method capable of learning from scarce, noisy data across a diverse array of domains. We highlight that VIGNs are able to (1) unify graph networks, ODEs, potential networks, and symplectic inductive biases for learning precise trajectories in large many-body systems, (2) make learning data-efficient, (3) maintain flexibility in learning from generalized momenta and easily extended to canonincal coordinates, and (4) build higher order variational integrators through partitioned Runge-Kutta methods. We note that different networks produce variable results in different settings, as would be expected. In spite of this, we see that potential-based networks are consistently the best at achieving the lowest state and energy MSE and, more importantly, that VIGN is always first or second among these methods. We believe this is an important outcome for practitioners, as VIGN can reliably be used across systems to attain excellent performance in terms of both state and energy error.

# 4

# Port-Hamiltonian Neural Networks for Learning Explicit Time-Dependent Dynamical Systems

# Abstract

Accurately learning the temporal behavior of dynamical systems requires models with well-chosen learning biases. Recent innovations embed the Hamiltonian and Lagrangian formalisms into neural networks and demonstrate a significant improvement over other approaches in predicting trajectories of physical systems. These methods generally tackle autonomous systems that depend implicitly on time or systems for which a control signal is known a priori. Despite this success, many real world dynamical systems are non-autonomous, driven by time-dependent forces and experience energy dissipation. In this study, we address the challenge of learning from such non-autonomous systems by embedding the port-Hamiltonian formalism into neural networks, a versatile framework that can capture energy dissipation and time-dependent control forces. We show that the proposed *port-Hamiltonian neural network* can efficiently learn the dynamics of nonlinear physical systems of practical interest and accurately recover the underlying stationary Hamiltonian, time-dependent force, and dissipative coefficient. A promising outcome of our network is its ability to learn and predict chaotic systems such as the Duffing equation, for which the trajectories are typically hard to learn.

## 4.1 Introduction

Neural networks (NNs), as universal function approximators (Hornik et al. 1989), have shown resounding success across a host of domains including image segmentation (He, Gkioxari, et al. 2017), machine translation (Devlin et al. 2019), and material property predictions (Yao et al. 2018; Toussaint et al. 2018). However, their performance in learning and generalizing the long-term behaviour of dynamic systems governed by known physical laws from state data has often been limited (Pukrittayakamee et al. 2009; Samuel Greydanus et al. 2019). New research in *scientific machine learning*, a field that tackles scientific problems with domain-specific machine learning methods, is paving a way to address this challenge. Concretely, it has been shown that physically-informed learning biases embedded in networks, such as Hamiltonian mechanics (Samuel Greydanus et al. 2019; Mattheakis, Sondak, et al. 2020), Lagrangians (Cranmer, Sam Greydanus, et al. 2020; Lutter et al. 2019), Ordinary Differential Equations (ODEs) (R. T. Q. Chen, Rubanova, et al. 2018), physics-informed networks (Maziar Raissi et al. 2017; Choudhary et al. 2020), generative networks (Toth et al. 2020), and Graph Neural Networks (Battaglia, Pascanu, et al. 2016; Sanchez-Gonzalez, Victor Bapst, et al. 2019) can significantly improve learning and generalization over vanilla neural networks in complex physical domains. The performance uplift arises primarily because learning biases are able to constrain networks to learn physically meaningful representations from data that are crucial to generalization.

Despite extensive research on learning biases, there is yet no method that accounts for non-autonomous systems i.e. systems with explicit time dependence. Non-autonomous dynamics feature prominently in settings with externally driven or controlled time-dependent forces as well as in systems with energy dissipation, for example, interacting materials, forced oscillators and charge-discharge cycles. Defining a network to accurately learn and predict the dynamics of such systems from position and momentum data is therefore of critical practical interest. We address this challenge by embedding the port-Hamiltonian formalism (Schaft 2007; Ortega et al. 2002; Acosta et al. 2005; Zheng et al. 2018; Cherifi 2020) into neural

networks. We show that the structure of this formulation can be used to uncover the underlying Hamiltonian, force and damping terms given position and momentum data and as such, can be used to accurately predict the long-range trajectories of many forced/damped systems. We extensively benchmark our network on a range of tasks including a simple mass-spring system with damping and external force, a Duffing system in both the non-chaotic and chaotic regimes, and a relativistic Duffing system. Our proposed network consistently outperforms other approaches while accurately recovering both the driving force and the damping coefficient. Furthermore, using minimal data, we show that our network can visually recover the Poincaré section of the Duffing system in a chaotic regime, emphasizing how our network can be used to identify and understand chaotic trajectories.

## 4.2 Background

### 4.2.1 Hamiltonian Neural Networks

Recently, the authors of (Samuel Greydanus et al. 2019) demonstrated that the dynamics of an energy conserving autonomous system can be accurately learned by guiding a neural network to predict a Hamiltonian - an important representation of a dynamical system. Considering a dynamical system of $M$ objects, the Hamiltonian $\mathcal{H}$ is a scalar function of a position vector $\mathbf{q}(t) = (q_1(t), q_2(t), ...., q_M(t))$ and momentum vector $\mathbf{p}(t) = (p_1(t), p_2(t), ...., p_M(t))$ that obeys Hamilton's equations,

$$\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \qquad \dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}, \tag{4.1}$$

where $\dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt}$ and $\dot{\mathbf{p}} = \frac{d\mathbf{p}}{dt}$.

Using the Hamiltonian formalism, (Samuel Greydanus et al. 2019) showed that a NN with parameters $\theta$ can be used to learn a Hamiltonian $\mathcal{H}_\theta(\mathbf{q}, \mathbf{p})$ given $\mathbf{q}$ and $\mathbf{p}$ as inputs to the network. The time derivatives are recovered from Eq. (4.1) by differentiating $\mathcal{H}_\theta$ with respect to its inputs using automatic differentiation. The resulting system has the form,

$$\dot{\mathbf{x}} = s(\mathbf{x}), \tag{4.2}$$

where $\mathbf{x} = (\mathbf{q}, \mathbf{p})$ and $s$ is determined by Eq. (4.1) after differentiating the trained NN. Eq. (4.2) can be discretized using a time-integrator to determine the trajectory of an initial state. Moreover, since $s$ is the symplectic gradient of the Hamiltonian, energy conservation is embedded into the method by construction. A symplectic map is a smooth manifold that generalises the phase space of classical systems by defining vector fields along which total energy can be conserved. That is to say, symplectic maps preserve the energy of a classical system as they are traversed. A symplectic integrator is an integrator whose solution resides on a symplectic manifold. Given these advantages, Hamiltonian Neural Networks (HNNs) outperform traditional approaches that directly predict the state time derivatives from the input state. In some settings, it is more advantageous to use the Lagrangian formalism of classical mechanics. Fortunately, the Hamiltonian and Lagrangian are fundamentally connected via a Legendre transform that helps us transform coordinates from generalised momenta to canonical momenta. As a consequence of the transform, the Hamiltonian is the sum of kinetic and potential energies while the Lagrangian is the difference. However, the standard Hamiltonian formulation does not readily generalize to damped or forced time-varying systems.

### 4.2.2 Port-Hamiltonian framework

The port-Hamiltonian (Schaft 2007; Ortega et al. 2002; Acosta et al. 2005; Zheng et al. 2018; Cherifi 2020) is a well studied formalism that generalizes Hamilton's equations to incorporate energy dissipation and an external control input to a dynamical system. Hamilton's equations in the port-Hamiltonian framework are represented as:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{p}} \end{bmatrix} = \left( \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} + \mathbf{D}\left(\mathbf{q}\right) \right) \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial \mathbf{q}} \\ \frac{\partial \mathcal{H}}{\partial \mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{G}\left(\mathbf{q}\right) \end{bmatrix} \mathbf{u}, \tag{4.3}$$

where $\mathbf{D}(\mathbf{q})$ is a damping matrix, $\mathbf{u}$ is a temporal control input, $\mathbf{G}(\mathbf{q})$ is a non-linear scaling of the position vector, $\mathbf{I}$ is the identity matrix, and $\mathbf{0}$ the zero matrix. The damping matrix is semi-positive definite. This general formalism readily reduces to

the standard Hamiltonian system when $\mathbf{D} = \mathbf{0}$ and $\mathbf{u} = \mathbf{0}$. The port Hamiltonian has been used in control applications where explicit knowledge of the control term $\mathbf{u}$ is known and was recently shown to reveal promising results in NNs (Zhong, Dey, et al. 2020b). Note that in (Zhong, Dey, et al. 2020b), the vector $[\mathbf{q}, \mathbf{p}, \mathbf{u}]$ is provided as input to the network. However, in many applications the control force $\mathbf{u}$ is unknown and it is therefore of interest to uncover the underlying forcing term from the data of the state vector, where no explicit knowledge of the control input and damping term are available.

### 4.2.3 Related Work

While Hamiltonian mechanics presents one way to address learning dynamical systems, numerous recent methods highlight how incorporating other physically-informed inductive biases into neural networks can improve learning.

Functional priors, for example, embed the full functional form of an equation into the NN. Physics-informed neural networks (PINNs) (Maziar Raissi et al. 2017; M. Raissi et al. 2019) and Hamiltonian networks (Mattheakis, Sondak, et al. 2020) are two such approaches that look at directly embedding the equations of motion into the loss function. While PINNs are data-driven approaches that rely on Autograd (Maclaurin et al. 2015) to compute partial-derivatives of a hidden state, Hamiltonian networks are data-independent approaches that pre-specify the full functional form of a system-specific Hamiltonian in the loss function.

Many recent methods have also sought to embed integrators into the training process. Indeed doing so induces an effectively continuous depth neural network able to perform large time-step predictions. NeuralODE (R. T. Q. Chen, Rubanova, et al. 2018) presents one way to tackle back-propagating through this continuous depth network more efficiently.

More recent work has looked at generalizing this approach to different and more complex data structures and topologies that standard NeuralODEs cannot represent (Dupont et al. 2019). Other work, such as (Zhu et al. 2020), theoretically

shows the importance of using symplectic integrators over Runge-Kutta methods to evolve Hamiltonian systems in NNs.

In (Battaglia, Pascanu, et al. 2016), the authors detailed how a Graph Neural Network, designed to capture the relational structure of systems, can be used to learn dynamics of interacting particles. This work has been exploited in numerous advances (Sanchez-Gonzalez, Godwin, et al. 2020; Sanchez-Gonzalez, Heess, et al. 2018; Cranmer, Sam Greydanus, et al. 2020) and emphasizes how a relational inductive bias can significantly improve learning.

In (Cranmer, Sam Greydanus, et al. 2020) and (Samuel Greydanus et al. 2019) it has been shown that by learning the Hamiltonian or the Lagrangian of a system, it is possible to accurately predict the temporal dynamics and conserve energy. The work of (Lutter et al. 2019) also showed that by exploiting the Euler-Lagrange equation, it is possible to predict a controlled double pendulum - a system pertinent for controlled robots. A recent advance shows that Hamiltonian and Lagrangian NNs can be drastically improved if they are optimized over Cartesian coordinates with holonomic constraints (Finzi, Stanton, et al. 2020).

Despite the significant breakthroughs, there is no existing method that investigates explicit time-dependence and damping in dynamical systems, two elements that are often found in real world problems. As such, we outline a novel technique to address this challenge.

## 4.3 Method

### 4.3.1 Theory

In this section we introduce port-Hamiltonian Neural Networks (pHNNs). We begin by illustrating how a NN takes the form of the port-Hamiltonian formulation of Eq. (4.3) with two modifications. First, our approach exploits the fact that many damped systems consist of a non-zero, state-independent damping term in the lower right quadrant, so we replace the damping matrix $\mathbf{D}(\mathbf{q})$ with a state independent matrix for which only the lower right term is non-zero and represented

by $\mathbf{N}$. Secondly, in order to generalize to time dependent forcing, we replace $\mathbf{G}(\mathbf{q})\mathbf{u}$ with the force field $\mathbf{F}(t)$. The resulting representation,

$$\begin{bmatrix}\dot{\mathbf{q}}\\ \dot{\mathbf{p}}\end{bmatrix} = \left(\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ -\mathbf{I} & \mathbf{0}\end{bmatrix} + \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{N}\end{bmatrix}\right)\begin{bmatrix}\frac{\partial \mathcal{H}}{\partial \mathbf{q}}\\ \frac{\partial \mathcal{H}}{\partial \mathbf{p}}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{F}(t)\end{bmatrix}, \tag{4.4}$$

is general enough to handle many well-known forced systems but also specific enough to tackle learning in physical domains of practical importance such as the Duffing equation.

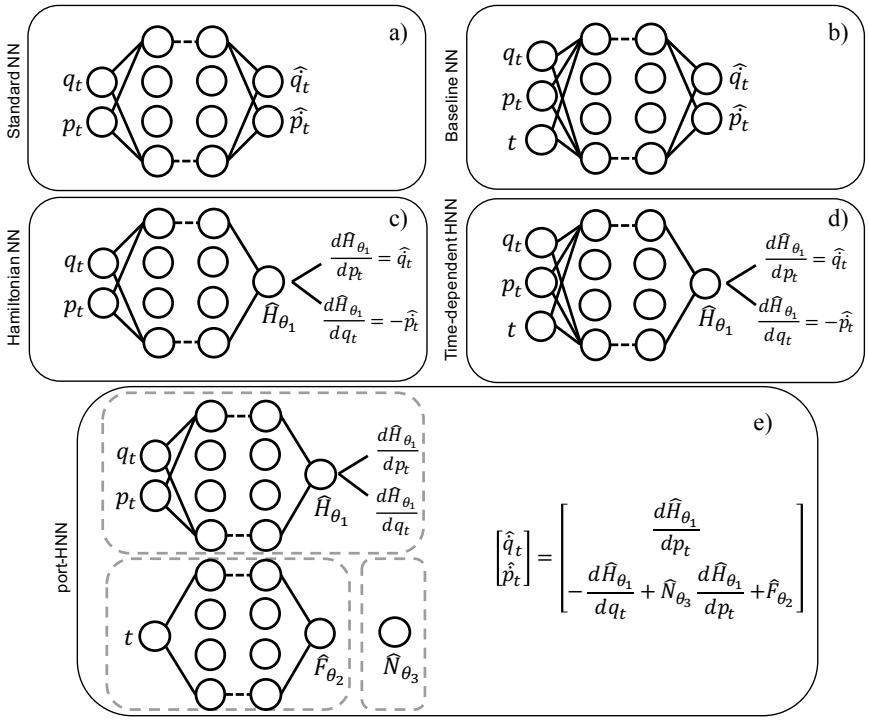



**Figure 4.1:** Architectures used in this study to learn dynamical systems. The naive extension of a standard feed forward NN (outlined in (a)) to incorporate time as an additional variable is shown in (b) and considered as the baseline network. The standard HNN in (c) is extended to receive time as an input and demonstrated by (d). Our innovation is presented in (e), which exploits port-Hamiltonians and explicitly learns the force $F_{\theta_2}$, the damping term $N_{\theta_3}$ and the Hamiltonian $\mathcal{H}_{\theta_1}$ to predict the state-time derivatives.

The architecture for the proposed pHNN model and a summary of comparable existing approaches are shown in Fig. 4.1. A standard feed forward NN in Fig. 4.1(a) and an HNN in Fig. 4.1(c) (Samuel Greydanus et al. 2019) take $q$ and $p$ as inputs and are trained to yield the time derivatives of the input, with the HNN learning

an intermediate Hamiltonian and employing backpropagation to compute the final output. A natural way to extend these architectures for time-varying non-autonomous systems is to include time as an additional input. This gives rise to the baseline network (Baseline NN) represented by Fig. 4.1(b) and a time-dependent HNN (TDHNN) shown by Fig. 4.1(d).

Although the Baseline NN and TDHNN incorporate time, they do not provide information for the underlying dynamics of the system. On the other hand, the pHNN is able to extract and provide information about the stationary Hamiltonian, the driving force, and the damping term. Moreover, the pHNN consistently outperforms all the network architectures shown in Figs. 4.1(a-d) across the applications investigated in this study.

### 4.3.2 Network optimization

The training of the pHNN consists of feeding the inputs $[\mathbf{q}, \mathbf{p}, t]$ into the model. The first component, $\mathcal{H}_{\theta_1}$, consists of three hidden layers designed to predict a stationary $\hat{\mathcal{H}_{\theta_1}}$ from $[\mathbf{q}, \mathbf{p}]$ input data. The second component, $\hat{\mathbf{N}}_{\theta_3}$, consists of a single weight parameter (i.e. $\theta_3$ is a single node) designed to learn the damping. The third neural-network unit $\hat{\mathbf{F}}_{\theta_2}$ solely depends on $t$ and consists of three hidden layers designed to predict a time-varying force. The output of each component is transformed through Eq. (4.4) to obtain predictions of the state time derivatives $[\hat{\dot{\mathbf{q}}}, \hat{\dot{\mathbf{p}}}]$. Using these predicted quantities we construct the loss function for optimizing the pHNN:

$$\mathcal{L} = ||\hat{\dot{\mathbf{q}}}_t - \dot{\mathbf{q}}_t||_2^2 + ||\hat{\dot{\mathbf{p}}}_t - \dot{\mathbf{p}}_t||_2^2 + \lambda_F||\hat{\mathbf{F}}_{\theta_2}||_1 + \lambda_N||\hat{\mathbf{N}}_{\theta_3}||_1, \tag{4.5}$$

where $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$ are known ground truth data. The first two components of the left hand side of Eq. (4.5) minimize the difference between the predicted and ground truth state time derivatives with a squared error loss. The last two components in Eq. (4.5) are the forcing and damping terms that are added to the loss function with an $L_1$ penalty when using pHNN. Using an $L_1$ penalty on these terms encourages the network to learn simpler models. We empirically found that this technique prevents the pHNN from learning spurious force and damping terms in unforced

and undamped systems compared to an $L_2$ penalty. The regularization parameters $\lambda_F$ and $\lambda_N$ were determined via grid search (see Appendix C.1). We use 200 nodes per hidden layer and find that most activation functions, including $\tanh(\cdot)$, $\sin(\cdot)$ and $\cos(\cdot)$ yield comparable results. We use Adam optimizer with a learning rate of $10^{-3}$ for all experiments.

We generate our data using a RK-4 integrator given some initial conditions for each system. We use a small $\Delta t = 0.001$ to evaluate the integral and ensure ground truth data is generated with rtol $= 10^{-10}$. The gradients of the state at each integration step are computed using the underlying differential equation. Details about the sampling of the initial conditions are described independently for each system in section 4.4.

We note that in some settings it might be hard to obtain the ground truth state time-derivatives $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$ for training. A natural way to address this problem is to embed an integrator into the training, similar to NeuralODE (R. T. Q. Chen, Rubanova, et al. 2018). As such, we also run our study with an embedded RK-4 integrator. Our method is still the most performant when all the methods incorporate an embedded RK-4 integrator (see Appendix C.4) and the loss function is $([\hat{q}, \hat{p}]_{t+1} - [q, p]_{t+1})^2$ . In other words, our system can learn from either state time derivative data or directly from state data given an embedded integrator. We were motivated to use gradient data since HNN is trained in this way and we wanted a fair comparison. Code is made available here [1].

### 4.3.3 Testing

Once trained, each of the networks in Fig. 4.1 can approximate $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$. As such, these networks can be used in a time integrator to evolve initial conditions in the test set. We refer to the integration for $t \in (0, T_{\max})$ as the *state rollout*. We measure the performance of the network by comparing the predicted state rollout with the ground truth. Specifically, we assess the networks performance by computing the

[1] https://github.com/shaandesai1/PortHNN

mean squared error (MSE) of the predicted state variables and predicted energy (Hamiltonian) across the integration time,

$$\text{MSE}_{\text{state}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{q}_i - \hat{\mathbf{q}}_i)^2 + \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i - \hat{\mathbf{p}}_i)^2 \tag{4.6}$$

$$\text{MSE}_{\text{energy}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{H}(\mathbf{q}_i, \mathbf{p}_i) - \mathcal{H}(\hat{\mathbf{q}}_i, \hat{\mathbf{p}}_i)\right)^2, \tag{4.7}$$

where $N = T_{\text{max}}/\Delta t$, with $\Delta t$ the time step size. These terms are computed for multiple initial conditions during inference and averaged across them. A guide to the training pipeline is outlined in Appendix C.2.

## 4.4 Results

We benchmark the performance of pHNNs against the other networks shown in Fig. 4.1. We evaluate the methods over datasets that cover simple time-independent systems to complex chaotic damped and driven dynamical systems. The results are presented in order of increasing complexity from a model perspective.

### 4.4.1 Simple Mass-Spring System

We begin our analysis with a simple mass-spring system (harmonic oscillator), obeying Hooke's Law from classical physics with no force or damping. The Hamiltonian that describes such a system reads:

$$\mathcal{H} = \frac{1}{2}kq^2 + \frac{p^2}{2m}, \tag{4.8}$$

where $k$ is the spring constant and $m$ denotes the mass. In this one-dimensional system the position and momentum are scalar functions of time.

**Training:** Without loss of generalization, we set $k = m = 1$ for our experiments. We randomly sample 25 initial training conditions $[q_0, p_0]$ that satisfy $q_0^2 + p_0^2 = r_0^2$ where $1 \leq r_0 \leq 4.5$ which corresponds to sampling initial conditions with energies in the range $[1, 4.5]$. We evolve each initial state using a RK4 integrator with $\Delta t = 0.05$ and $T_{\text{max}} = 3.05$.

**Testing:** We evaluate the performance of the NNs by sampling 25 random initial conditions in the same way as training. We investigate the simple harmonic oscillator system and show that learning a separate, regularized forcing term results in better state and energy predictions in comparison to TDHNN and the Baseline NN. Learning a separate forcing term and regularizing pHNN keeps the time component independent of the Hamiltonian and therefore allows us to closely match the performance of the standard time-implicit HNN.

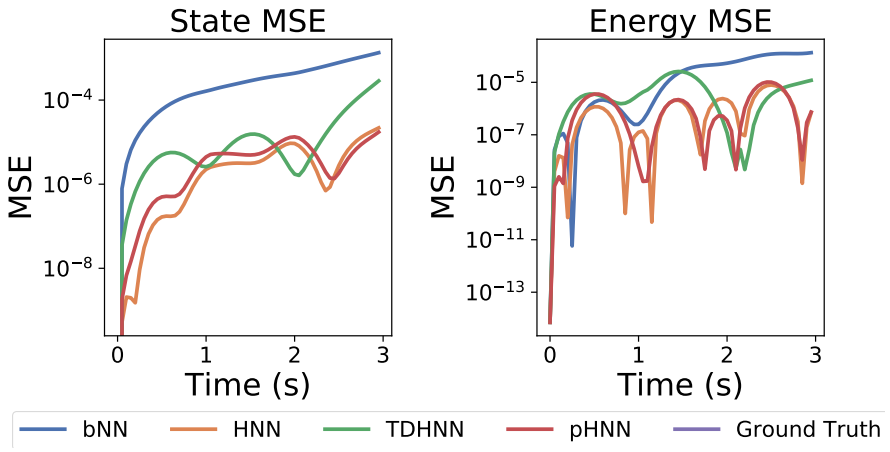
In particular, in Fig. 4.2(a) we show the state and energy MSE for an initial state from the testing set. Fig. 4.2(b) presents the predicted force (blue solid line) and damping over time $\nu\frac{\partial H}{\partial p}$ (red solid curve), while black dots corresponds to ground truth observations. We observe that the error in the predictions is of the order of $10^{-5}$ for recovering the force function and of $10^{-8}$ for the damping term. In Fig. 4.2(c) we report the state and energy MSE averaged along all the testing initial states, where the black lines in the histograms represent the error bars of the statistics. In Fig. 4.2(d) we highlight the learnt Hamiltonian from the randomly sampled initial condition of (a) and (b).

### 4.4.2 Damped Mass-Spring System

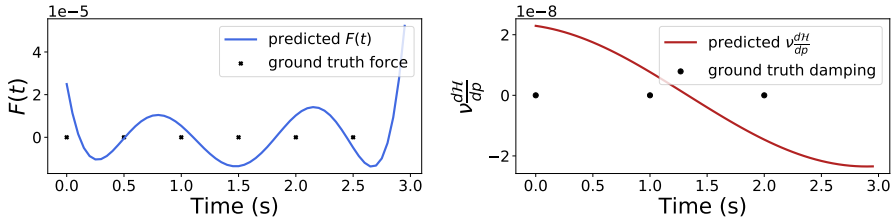
We extend the simple mass-spring system to include a damping term that reduces the initial energy of the system over time. The inclusion of this term violates energy conservation and therefore we cannot write a scalar Hamiltonian for such a system (see Appendix C.3 for details). Knowing this a priori already gives us an indication that the HNN will perform poorly on such a system.

**Training:** We have 20 initial training conditions, with position and momentum uniformly sampled in $[-1, 1]^2$. Each trajectory is evolved until $T_{\max} = 30.1$ with a $\Delta t = 0.1$. We fix the damping coefficient $\nu = 0.3$ without loss of generality.

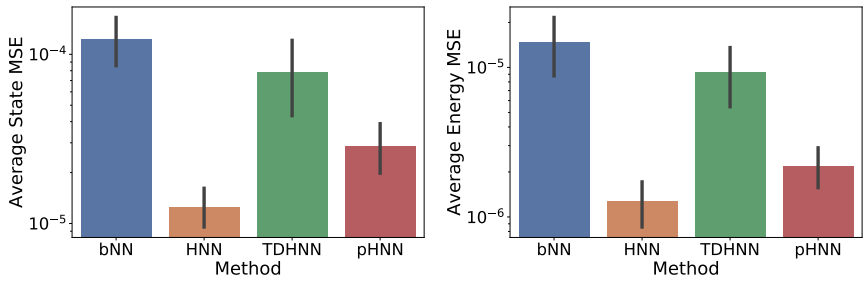
**Testing:** At inference, we compute the average rollout MSE of 25 unseen initial conditions sampled in the same manner as the training data. We show the state and energy MSE for an initial state from the testing set in Fig. 4.3(a). Figure 4.3(b) outlines the predicted force and damping for an arbitrary initial state. In



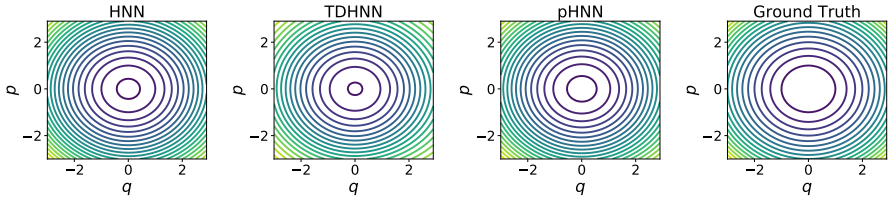

**(a)** State and energy MSE as a function of time of an initial condition in the test set.




**(b)** Learnt force and damping terms by pHNN



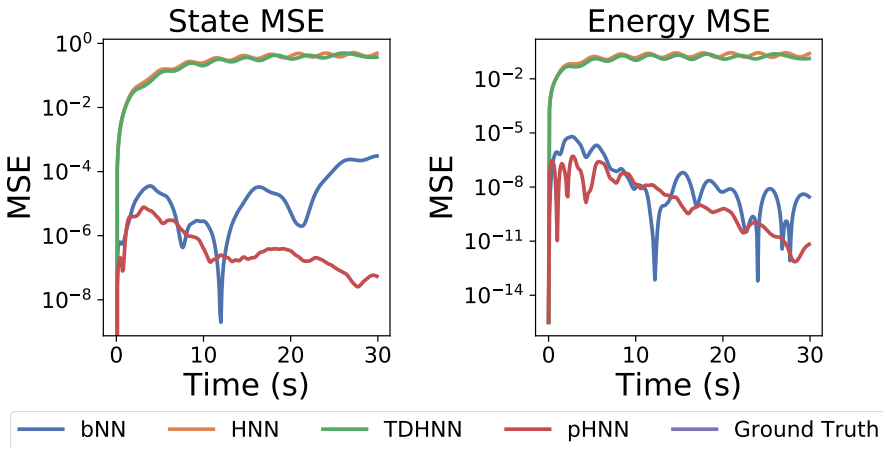

**(c)** State and energy MSE averaged across 25 initial test states (error bars showing $\pm 1\sigma$).



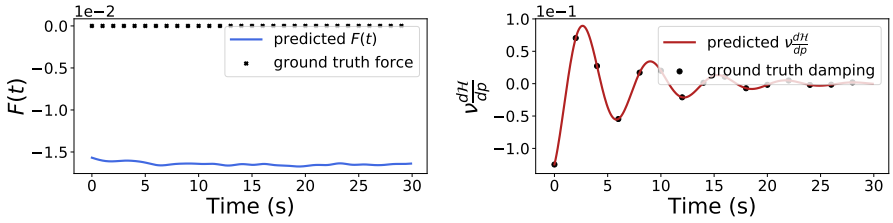

**(d)** The learnt Hamiltonian across methods

**Figure 4.2:** The simple mass-spring system has no explicit time dependence. We see that the pHNN can almost recover the dynamics as well as the HNN. While the pHNN does learn a non-zero force and damping term, their contribution to $\frac{dp}{dt}$ is small.

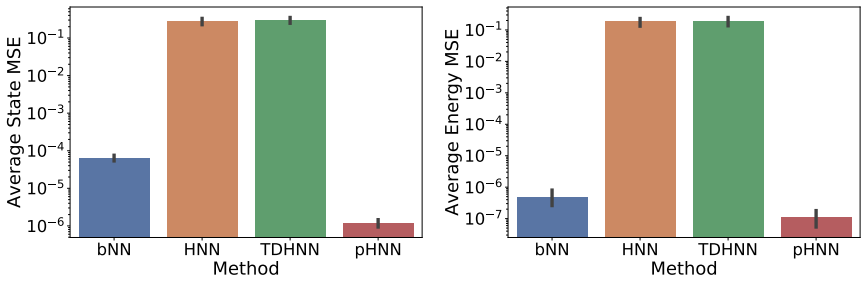
Fig. 4.3(c) we report the average state and energy rollout MSE and in Fig. 4.3(d) we show the learnt Hamiltonian.



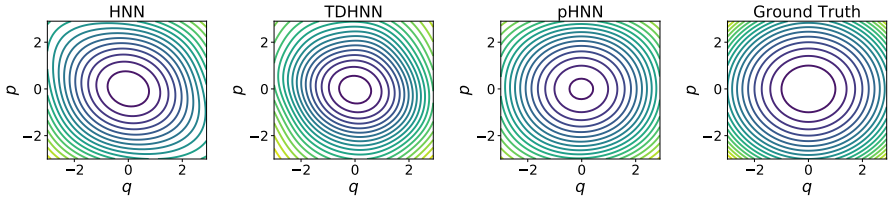

**(a)** State and energy MSE as a function of time of an initial condition in the test set.




**(b)** Learnt force and damping terms by pHNN




**(c)** State and energy MSE averaged across 25 initial test states (error bars showing $\pm 1\sigma$).




**(d)** The learnt Hamiltonian across methods

**Figure 4.3:** Damped mass-spring setting: The baseline NN and pHNN recover the underlying dynamics well. pHNN is also able to accurately learn the damping coefficient since the predicted damping is indistinguishable from the ground truth.

We observe in Fig. 4.3 that both baseline NN and pHNN recover the dynamics well, whereas the HNN (as expected) and TDHNN struggle to learn the dynamics of

the damped system. This failure happens because there is no direct way of writing a scalar Hamiltonian with damping and thus, both the HNN and TDHNN cannot learn the underlying dynamics. This observation indicates that a partial inductive bias is not enough to make a network robust at predicting the dynamics well. A second observation shows that the pHNN learns a non-zero oscillating force. That implies a possible leaking of the information from the predicted Hamiltonian into the predicted force and vice versa. This arises because there is an identifiability challenge inferring the damping and force exclusively from the state data we provide. In spite of this challenge, we find that the pHNN converges to forcing and damping terms consistent with the ground truth generating terms and sufficient to inform us about the underlying dynamics as well as to evolve initial states with small numerical error.

### 4.4.3 Forced Mass-Spring System

We complete the investigation of the simple mass-spring problem by including a driven time-dependent force that controls the system. To understand the effect of the force we consider an undamped driven oscillator system. Typically, while we cannot write the Hamiltonian for a damped system, we can write one for a forced system. We study two cases of forced mass-spring systems. The first has the following Hamiltonian form:

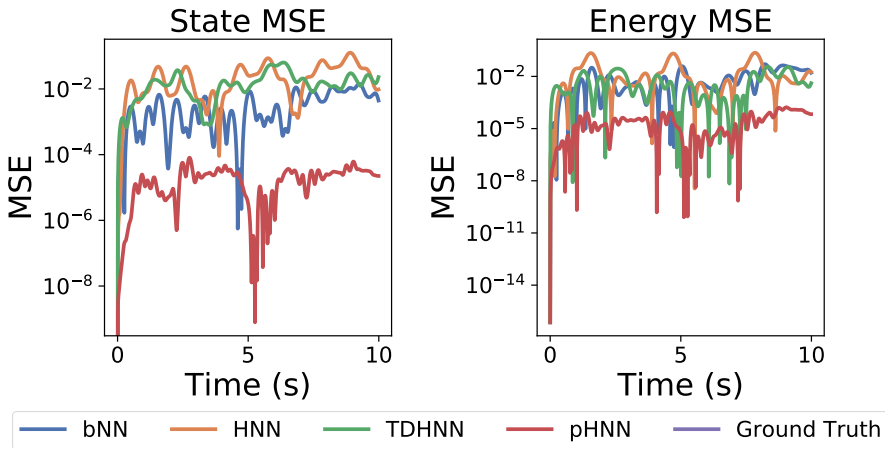
$$\mathcal{H} = \frac{1}{2}kq^2 + \frac{p^2}{2m} - qF_0 \sin(\omega t). \tag{4.9}$$

The second has a more complex force described by the Hamiltonian:

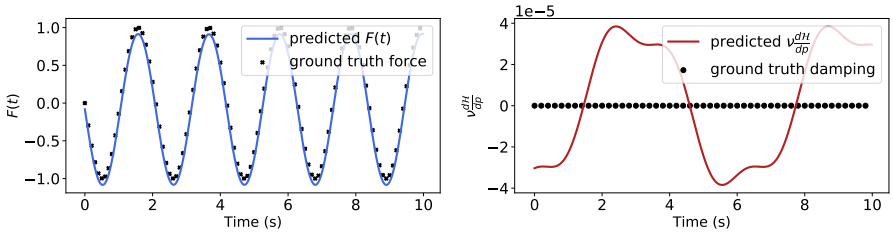
$$\mathcal{H} = \frac{1}{2}kq^2 + \frac{p^2}{2m} - qF_0 \sin(\omega t)\sin(2\omega t), \tag{4.10}$$

where $F_0$ and $\omega$ is the amplitude and frequency of the external force term. The forced mass-spring system is typically used to study resonance effects, e.g. in material science, and plays an important role in a wide range of applications including music, bridge design, and molecular excitation making it an important system to investigate.

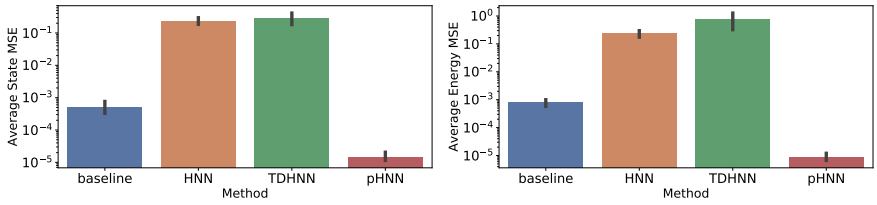
**Training:** In both systems of Hamiltonian Eqs. (4.9) and (4.10), we use 20 initial conditions, where the initial state $[q_0, p_0]$ is sampled such that $q_0^2 + p_0^2 = r_0^2$ where $1 \leq$



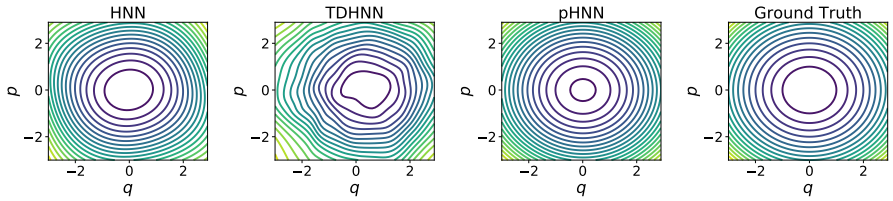

**(a)** State and energy MSE as a function of time of an initial condition in the test set.




**(b)** Learnt force and damping terms by pHNN.




**(c)** Rollout state and energy MSE averaged across 25 initial test states (error bars showing $\pm 1\sigma$).

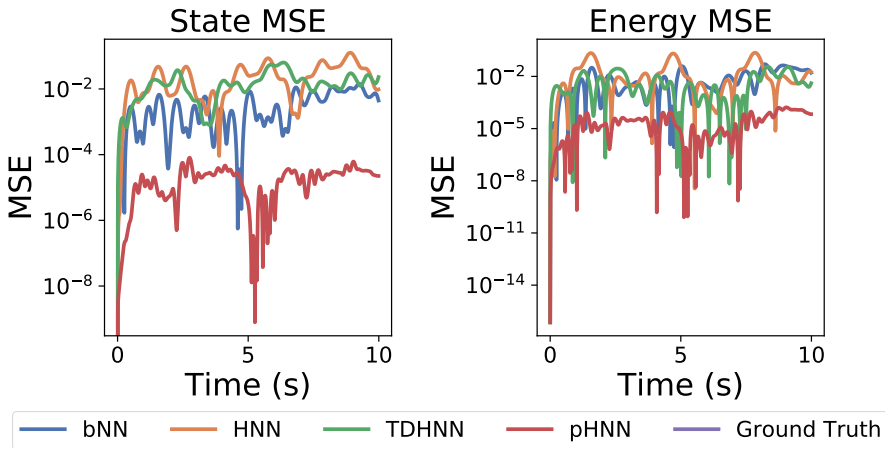



**(d)** The learnt Hamiltonian across methods

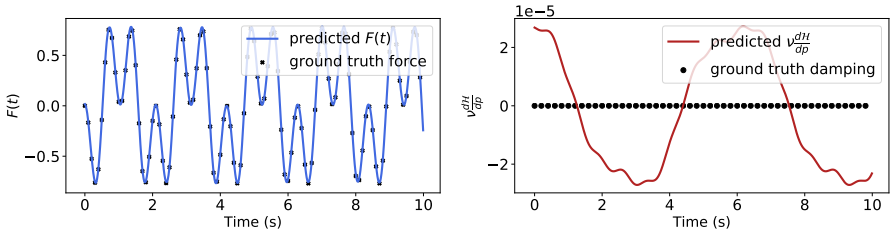
**Figure 4.4:** Forced mass-spring of Eq. (4.9): Standard HNN cannot learn the underlying dynamics as it has no explicit-time dependence. pHNN shows the best performance as it explicitly learns a time-dependent force.

$r_0 \leq 4.5$. For the force term we set $F_0 = 1$ and $\omega = 3$, and without loss of generality we set $k = m = 1$. The states are rolled out to $T_{\text{max}} = 10.01$ at a $\Delta t = 0.01$.

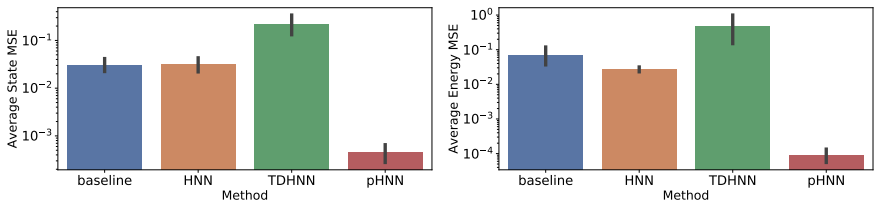
**Testing:** At inference, we compute the rollout of 25 unseen initial conditions in



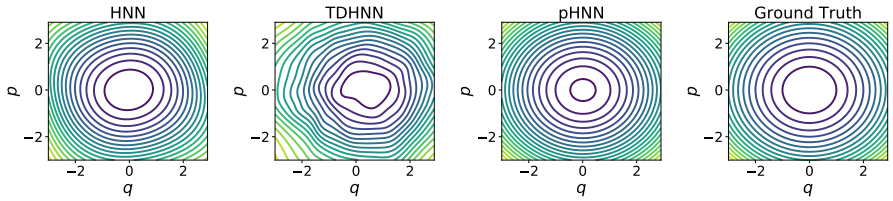

**(a)** State and energy MSE as a function of time of an initial condition in the test set.




**(b)** pHNN recovers force and damping.




**(c)** Rollout state and energy MSE averaged across 25 initial test states (error bars showing $\pm 1\sigma$).




**(d)** The learnt Hamiltonian across methods

**Figure 4.5:** Forced mass-spring system of Eq. (4.10): pHNN is able to recover a non-harmonic force and evolves testing initial states better than the other models.

the same range as the training data. Figures 4.4(b) and 4.5(b) demonstrate the predicted forces and dissipation terms for the systems of Eqs. (4.9) and (4.10), respectively. Accordingly, in Figs. 4.4(c) and 4.5(c) we report the average state and energy rollout MSE for each system.

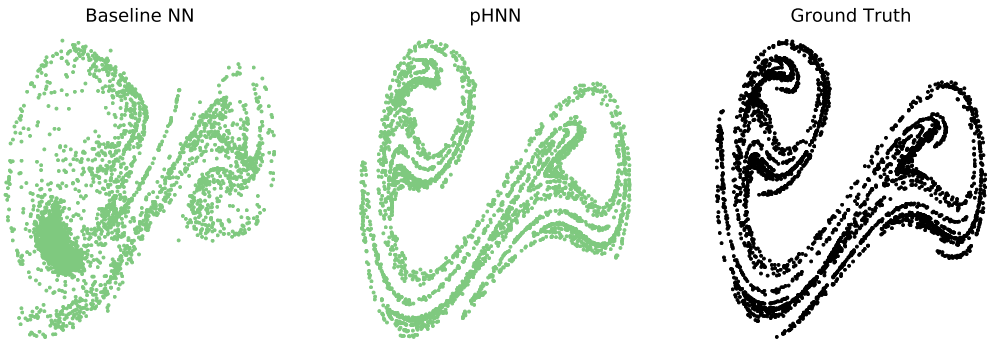



**Figure 4.6:** Poincaré sections of a chaotic Duffing oscillator. Baseline NN (left) and pHNN (middle) are trained for 20000 iterations with 2000 data points. The left and middle images indicate the predicted Poincaré map for a initial state not used in networks optimization. The pHNN significantly outperforms the baseline NN at recovering the ground truth Poincaré section (right).

We study both systems to illustrate that while the baseline NN performs relatively well in comparison to the pHNN when a simple force is considered such as in the system of Eq. (4.9), a more complex force significantly hurts its performance in terms of state/energy MSE shown by Fig. 4.5(c). More importantly, we read in Figs. 4.4(b) and 4.5(b) that for both systems pHNN can recover the ground truth force quite precisely while it learns very small spurious damping terms that do not significantly contribute to the dynamics; the contribution to $\frac{dp}{dt}$ term is of the order $10^{-5}$, which is practically negligible.

### 4.4.4 Duffing Equation

Another problem that we investigate is given by the Duffing equation, a nonlinear dynamical system that includes both forcing and damping. The unforced and undamped stationary Hamiltonian $\mathcal{H}_{\text{stat}}$ of the Duffing system is given by:

$$\mathcal{H}_{\text{stat}} = \frac{p^2}{2m} + \alpha\frac{q^2}{2} + \beta\frac{q^4}{4}. \tag{4.11}$$

Unlike the simple mass-spring system, the Duffing equation has an additional quadratic function of $q$ that makes the system non-harmonic. The shape of the potential function can be tuned to be a double-well or a single well based on the coefficients $\alpha$ and $\beta$. The general Duffing equation includes a time variant force

and a damping term proportional to $\partial \mathcal{H}_{\text{stat}}/\partial q$. Typically the Duffing nonlinear equation of motion is written as:

$$\ddot{q} = -\delta \dot{q} - \alpha q - \beta q^3 + \gamma \sin(\omega t). \tag{4.12}$$

Different combinations of parameters $\alpha, \beta, \delta, \gamma, \omega$ make the Duffing system either chaotic or non-chaotic. We study both regimes. The Duffing equation reveals numerous phenomena of practical importance including frequency hysteresis (e.g. in magnets), elasticity and chaos theory.

**Non-Chaotic regime**

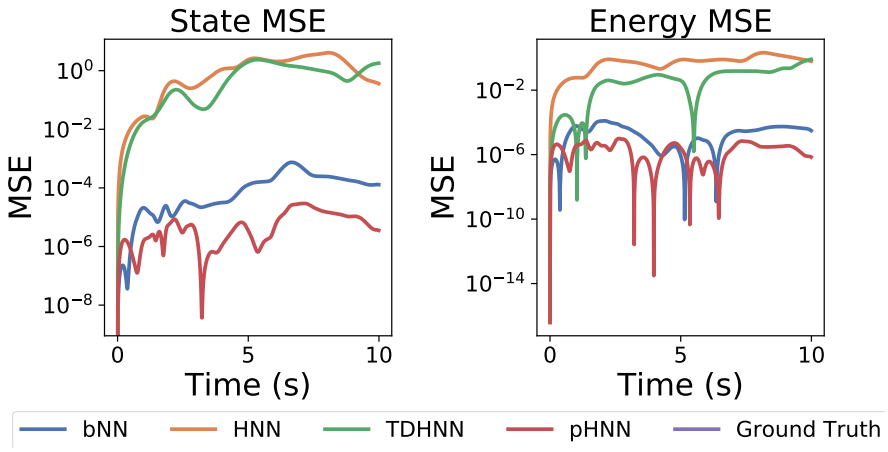
Given a set of initial parameters for the Duffing equation: $\alpha = -1, \beta = 1, \delta = 0.3, \gamma = 0.2, \omega = 1.2$ we can obtain training data in a non-chaotic regime of the Duffing system.

**Training:** We uniformly sample initial conditions in $[-1, 1]^2$ and use 25 initial conditions for training, rolled out to $T_{\text{max}} = 10.01$ with $\Delta t = 0.01$.

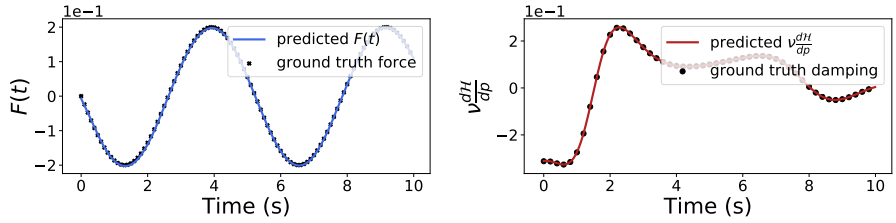
**Testing:** We integrate 25 unseen initial conditions at inference using the same $T_{\text{max}}$ and $\Delta t$ used to generate the training set. We evaluate all the neural network models on this testing set and present the results in Fig. 4.7. We observe in Fig. 4.7(b) that pHNN accurately recovers the underlying force and damping. Moreover, Fig. 4.7(c) indicates the pHNN outperforms the other models used in this study. We further assess the network performance by inspecting the predicted Hamiltonian. In Fig. 4.7(d) we outline the learnt $\mathcal{H}_{\text{stat}}$ as a function of $q$ and $p$ that comprise the phase space. We observe that pHNN can learn the functional form of $\mathcal{H}_{\text{stat}}$, outperforming the other architectures.

**Chaotic regime**

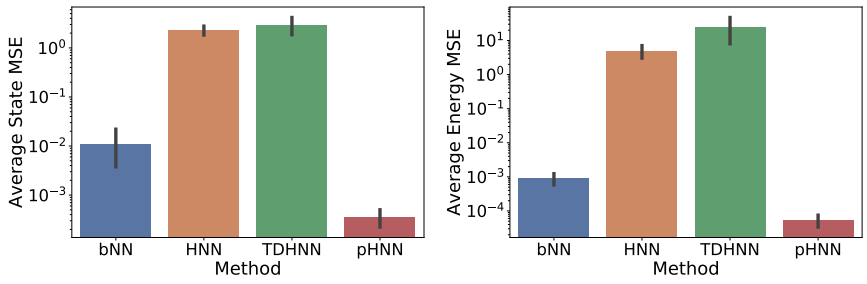
The choice of the parameters: $\alpha = 1, \beta = 1, \delta = 0.1, \gamma = 0.39, \omega = 1.4$, yields chaotic behavior in the Duffing system. Chaotic trajectories are highly sensitive to initial conditions and thus, it is much more difficult to learn from a chaotic system than from a non-chaotic.



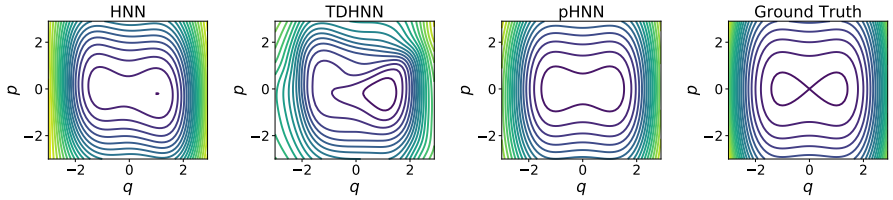

**(a)** State and energy MSE as a function of time of an initial condition in the test set.




**(b)** Learnt Force and Damping terms of pHNN




**(c)** State and energy MSE averaged across 25 initial test states (error bars showing $\pm 1\sigma$).

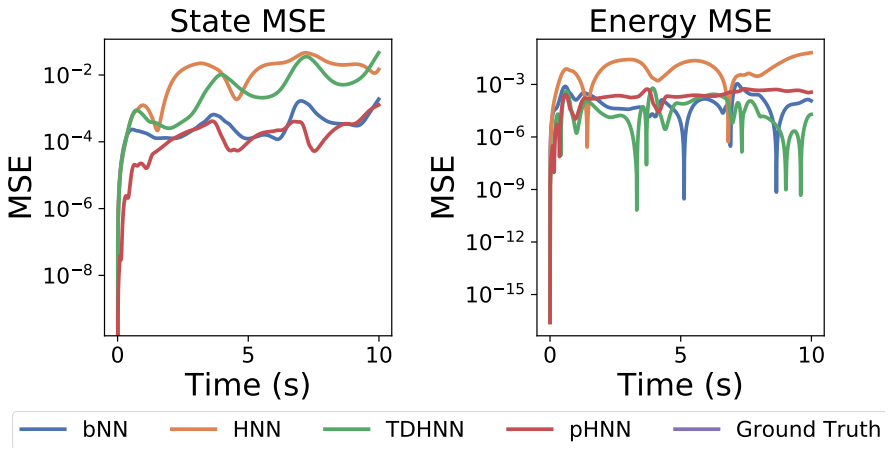



**(d)** The learnt Hamiltonian across methods

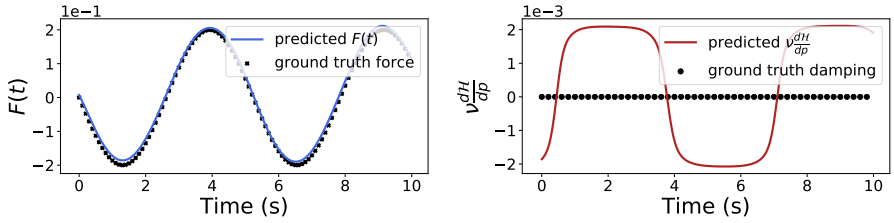
**Figure 4.7:** Duffing system (non-chaotic): pHNN significantly outperforms the other methods and is able to extract the ground truth force and damping coefficient.

**Training:** 20 initial conditions, sampled uniformly in $[-1, 1]^2$ each rolled out for one period $T = 2\pi/\omega$ where $\Delta t = T/100$ resulting in 2000 training points.

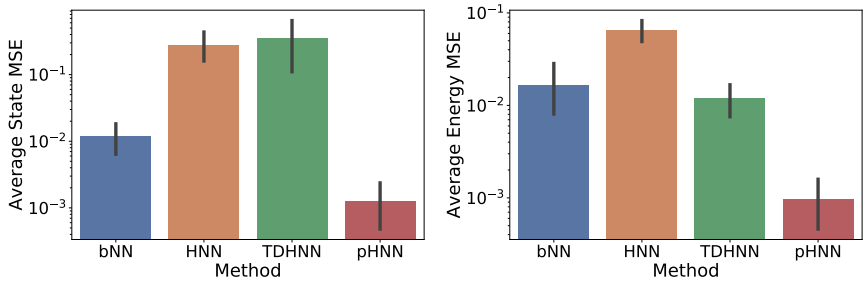
**Testing:** We test our system by assessing whether it is visually able to recover the



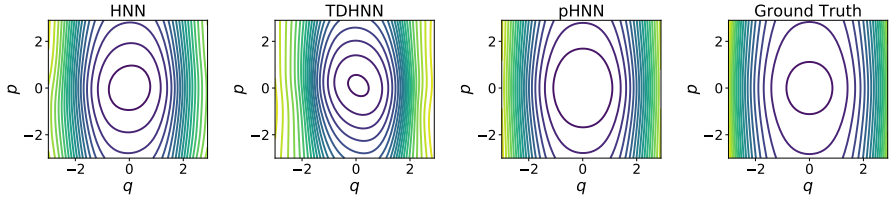

**(a)** State and energy MSE as a function of time of an initial condition in the test set.




**(b)** Learnt Force and damping terms of pHNN




**(c)** Rollout state and energy MSE averaged across 25 initial test states




**(d)** The learnt Hamiltonian across methods

**Figure 4.8:** Learned dynamics of a relativistic Duffing system.

ground truth Poincaré section of an initial condition and we focus our attention on baseline NN and pHNN since they are the most performant in the non-chaotic regime. The Poincaré map (or section) of a trajectory is measured by plotting the position and momentum values at regular intervals governed by the period of

the forcing term. For example, a simple mass-spring system will generate a single point in phase space when measured at regular intervals, while a chaotic system generates a more complex map. To visually assess the performance of our network through a Poincaré map, we test the system on a single initial condition, not used in the training set, rolled out to $T_{\max} = 18000$ with the same $\Delta t$ as in the training phase. In order to integrate our system to such a large $T_{\max}$ for this example we work under the assumption that we have explicit knowledge of the period of the force, and as such, we normalize the time variable with the period. We emphasize that while prior knowledge of $\omega$ assists the training, it is only used to extract the Poincaré map. This is necessary as the models are not explicitly trained on time steps beyond $2\pi/\omega$. The results are demonstrated in Fig. 4.6. In our study we find that the visual similarity between the section generated by pHNN is much closer to the ground truth than baseline. A simple quantitative measure of this similarity can be computed using the MSE between the 2D histogram plots. The pHNN MSE is 1.61 and bNN is 4.05. The outcome suggests that the pHNN can indeed be used to model chaos even after being trained with only a few data points from a chaotic trajectory. We believe this is a valuable result, since it implies that the pHNN can be used to model chaotic behavior.

### 4.4.5 Relativistic System

In the final experiment presented in this study, we go beyond non-relativistic classical mechanics and explore a different form of Hamiltonians. In particular, we investigate the motion of a driven relativistic particle in a nonlinear double well potential, which is mathematically represented by the Duffing equation in a relativistic framework. The Hamiltonian under consideration is:

$$\mathcal{H} = c\sqrt{p^2 + m_0^2 c^2} + \frac{\alpha}{2}q^2 + \frac{\beta}{4}q^4 - q\gamma\sin(\omega t), \tag{4.13}$$

where $c$ is the speed of light that typically is set to 1. For simplicity, we also set the rest mass $m_0 = 1$, though our framework naturally accounts for other values.

**Training.** We train on 25 initial conditions, uniformly sampled in $[0, 2]^2$. We consider $T_{\max} = 20.01$, $\Delta t = 0.01$, and the parameters $\alpha = 1, \beta = 1, \delta = 0, \gamma = 0.2, \omega = 1.2$.

**Testing.** Using the same parameters as training, we roll out 25 unseen initial conditions and present the results in Fig. 4.8. We observe that pHNN is able to recover the underlying force and outperforms the other architectures in generating the temporal state of previously unknown initial conditions. This experiment is evidence that pHNN can discover the dynamics of temporal systems independent of the form of the underlying Hamiltonian and consistently outperforms the other architectures investigated in this study.

## 4.5 Discussion

We have shown that pHNN outperforms other approaches in learning complex physical systems (tabularized results shown in Appendix C.4), as well as being able to recover the underlying stationary Hamiltonian, the external time varying force, and the damping term of non-autonomous systems. One challenge in achieving this result is fine-tuning the $\lambda_F$ and $\lambda_N$ regularisation coefficients for the force and damping. In the Duffing setting where we have both terms, it is possible to learn a shifted force i.e. $F = F_0 + \epsilon$. This is possible because the state-vectors $\mathbf{q}, \mathbf{p}$ do not provide enough information to simultaneously identify both the Hamiltonian and the force resulting in a leak of information between $\mathcal{H}$ and $F$. This is the reason why in some of the figures such as 4.2b and 4.4b, we see that we do not perfectly match the ground truth. Nevertheless, we find that a reasonable force and damping term are generally learnt which are sufficient to reveal the underlying dynamics.

While it may be argued that pHNN is constrained to learn and predict within the training time horizon, we believe our method is still versatile at informing us of periodic forcing, since we can inspect the force over time and, essentially, learn the period of the underlying force. This, in turn, can readily be used to renormalize the time variable at periodic intervals to integrate the system beyond the training time as we showed in the Poincaré map of Fig. 4.6.

We also run the entire set of experiments on noisy input state vectors where the noise is sampled from $\mathcal{N}(0, \sigma)$ with $\sigma \in [0.01, 0.1, 0.5]$. The details of the results can be found in Appendix C.5. We find that even with the addition of noise to the input state-vector, pHNN outperforms other methods.

While our study has focused extensively on understanding single particle systems, we also carry out a simple study on a 2 body coupled spring system where one of the masses is forced by a cosine varying signal. The results indicate that pHNN performs the best and has potential to scale to larger domains (see Appendix C.6).

## 4.6 Conclusion

We have shown that learning the dynamics of time-dependent non-autonomous systems can be achieved with pHNN, a versatile neural network embedded with the Port-Hamiltonian formulation. Our experimental investigation demonstrates that pHNN outperforms extensions of existing methods in numerous settings. Specifically, we outlined that the proposed network not only learns the underlying dynamics of a simple mass-spring system, achieving comparable performance to the efficient HNN architecture, but also extends to more complex nonlinear forced and damped physical systems. Furthermore, using pHNN we were able to show, with minimal training data, the ability to recover the Poincaré section of a chaotic driven system. Unlike existing methods, pHNN is able to identify systems where minimal state data is available and reveal the functional form of the controlling force, damping, and underlying stationary Hamiltonian. Collectively, these results form a strong basis for further advances in learning complex systems including, but not limited to, chemical bond forces, robotic motion, and more general controlled dynamics without explicit knowledge of the force and damping.

# 5

# One-Shot Transfer Learning of Physics-Informed Neural Networks

# Abstract

Solving differential equations efficiently and accurately sits at the heart of progress in many areas of scientific research, from classical dynamical systems to quantum mechanics. Recently, there has been a surge of interest in using Physics-Informed Neural Networks (PINNs) to tackle such problems as they provide numerous benefits over traditional numerical approaches. Despite numerous well-studied benefits for solving differential equations using neural networks, transfer learning has been under explored. In this chapter, we present a general framework for transfer learning PINNs that results in one-shot inference for linear systems of both ordinary and partial differential equations. This means that highly accurate solutions to many unknown differential equations can be obtained instantaneously without retraining an entire network. We demonstrate the efficacy of the proposed deep learning approach by solving several real-world problems, such as first- and second-order linear ordinary equations, the Poisson equation, and the time-dependent Schrödinger complex-value, partial differential equation.

## 5.1 Introduction

Differential equations appear in a broad array of domains, from infection models in biology (Kaxiras et al. 2020) to chaotic motion in physics (Choudhary et al. 2020). As such, efficiently and accurately solving these equations under various conditions continues to be an important and challenging problem in the scientific community. While traditional approaches such as Runge-Kutta (Dormand et al. 1987) and Finite Element Methods are well studied and provide solutions of high fidelity, recently, Physics-Informed Neural Networks (PINNs) (Karniadakis et al. 2021; M. Raissi et al. 2019) have attracted significant attention as an alternative framework for solving differential equations. PINNs are Neural Networks (NNs) that exploit backpropagation to compute partial derivatives and are thus capable of enforcing a known differential equation through a loss function. They have numerous advantages over traditional approaches such as eliminating the need for a numerical integrator, being able to generate continuous and differentiable solutions, improving accuracy in high dimensions, and maintaining memory efficiency (Karniadakis et al. 2021). However, a core benefit of using NNs to solve differential equations that remains under explored is transfer learning. This is particularly important for PINNs as they can be expensive to train from scratch. We therefore illustrate how the benefits of transfer learning, typically found in computer vision and natural language processing, naturally adapt to solving differential equations.

In this chapter, we show that a PINN pre-trained on a family of differential equations can be effectively re-used to solve new differential equations. Specifically, by freezing the hidden layers of a pre-trained network, we demonstrate that solving new differential equations reduces to optimizing/fine-tuning a linear layer. In doing so, we prove that, for linear systems of differential equations, the optimization is equivalent to solving the normal equations for a latent space of learnt functions. This implies that the optimal linear weights needed to satisfy a new differential equation can be computed in one-shot with the computational cost of a matrix inversion (see Appendix D.3). This therefore entirely eliminates the need for further training/fine-tuning, dramatically reducing the training overhead while

maintaining high solution accuracy. We investigate the efficiency of this approach by solving several ordinary differential equations (ODEs) as well as partial differential equations (PDEs) of practical interest. For many systems, we are able to identify highly accurate solutions to unseen differential equations in a fraction of the time needed to train the equations from scratch.

## 5.2 Background

The general form of an explicit $n^{\text{th}}$ order ODE can be written as:

$$F(t, \psi, \psi^{(1)}, ...., \psi^{(n-1)}) = \psi^{(n)}, \tag{5.1}$$

where $\psi^{(i)} = \frac{d^i\psi}{dt^i}$ is the $i^{\text{th}}$ derivative of the solution $\psi(t)$ with respect to the independent time variable $t$. Non-homogeneous linear ODEs, a subclass of the general form of Eq. 5.1, can be represented as follows:

$$\hat{D}_n\psi = f(t); \quad \hat{D}_n\psi = \sum_{i=0}^{n} a_i(t)\psi^{(i)}, \tag{5.2}$$

where $n$ denotes the order of the ODE, $f(t)$ is considered a forcing (or control) term that influences the homogeneity of the solution, and $a_i(t)$ is a time dependent coefficient for each derivative.

Traditionally, when an ODE is known *a priori*, the differential equation can be solved using integrators (such as Runge-Kutta) for given initial conditions (ICs). Recently, it has been shown that neural networks can be used to efficiently determine accurate solutions to such problems. As previously detailed, one such approach uses Physics-Informed Neural Networks (PINNs), whereby a neural network, with weights parametrized by $\theta$, is used to transform an input $t$ to output solutions $\psi_\theta(t)$. Then, by leveraging backpropagation and autograd (Maclaurin et al. 2015), exact derivatives of the network output can be computed with respect to the input $\frac{\partial^i\psi}{\partial t^i}$ where $i$ denotes the order of the derivative. Therefore, given ICs $u_{\text{ic}} = [\psi_0, \psi_0^{(1)}, .., \psi_0^{(n-1)}]^T$, and known differential operator $\hat{D}_n$ and force $f(t)$, the loss function of a PINN is defined as:

$$\mathcal{L} = (\hat{D}_n\psi_\theta(t) - f(t))^2 + (\bar{D}_0\psi_\theta(t) - \psi_{\text{ic}})^2, \tag{5.3}$$

where $\bar{D}_0\psi = [\psi(0), \psi^{(1)}(0), ..., \psi^{(n-1)}(0)]^T$. The first term enforces the differential equation and the second enforces the initial conditions. The network is then optimized using stochastic gradient descent to minimize the total loss.

Indeed, such a loss can be enforced for other architectures such as NeuralODE (R. T. Q. Chen, Rubanova, et al. 2018) and Reservoir Computing (Mattheakis, Joy, et al. 2021) which exploit recurrent neural networks. However, such networks are not as easily adaptable to PDEs as PINNs are. Many well known PDEs such as the diffusion, wave as well as Schrödinger equation, can be modeled using PINNs. With PDEs, additional variables are used as inputs $[t, x, y, ...]$, so the output is a function of these inputs $\psi(t, x, y, ..)$ subject to certain ICs and boundary conditions (BCs). As such, a loss function similar to Eq. 5.3 can be defined for PDEs.

The benefits of using a NN architecture to solve differential equations (both ODEs and PDEs) over traditional methods (such as Runge-Kutta or Finite Elements) include rapid inference, elimination of the curse of dimensionality, no accumulation of errors as would be found with integrators, continuously differentiable solutions, and low memory cost (Karniadakis et al. 2021). While these benefits have been extensively explored across a range of applications (H. Wang et al. 2021; Zhai et al. 2021a; Wolff et al. 2021; Sirignano et al. 2018; Mattheakis, Sondak, et al. 2020; McClenny et al. 2020), a limited study exists on how transfer learning techniques can be used (H. Wang et al. 2021; Mattheakis, Joy, et al. 2021; Guo et al. 2021). Here, we explore this idea extensively and identify a novel one-shot transfer learning framework for systems of linear differential equations that significantly speeds up inference on unseen differential equations.

## 5.3 Related Work

Constraining neural networks to learn solutions to differential equations was first introduced by Lagaris et.al (1998). The authors showed that partial derivatives of a neural network output with respect to its inputs can be analytically computed when the architecture of the network is known. Therefore, given the solution and its derivatives, it is possible to simultaneously enforce the underlying differential

equation as well as the ICs and BCs. Indeed this approach forms the basis of PINNs (Karniadakis et al. 2021; M. Raissi et al. 2019) where the analytic derivatives from Lagaris et.al (1998) are replaced with backpropagation, i.e. replacing the need for an analytic derivation of the partial derivatives. PINNs have since been extensively used across many applications including non-linear structures (R. Zhang et al. 2020), fluid flow on large domains (H. Wang et al. 2021), moving boundaries (S. Wang and Perdikaris 2021), inferring micro bubble dynamics (Zhai et al. 2021a), cardiac activation mapping (Sahli Costabal et al. 2020), ocean modelling (Wolff et al. 2021), bundle solvers (Flamant et al. 2020) and stochastic and high-dimensional PDEs (Karniadakis et al. 2021; Yang et al. 2018; Sirignano et al. 2018).

Recently, this technique has been extensively used to learn underlying dynamics from data - a concept first proposed by Howse et.al (1996). For example, numerous works show that energy conserving trajectories can be effectively learnt from data by enforcing known energy constraints such as Hamiltonians (Sanchez-Gonzalez, Victor Bapst, et al. 2019; Samuel Greydanus et al. 2019), Lagrangians (Cranmer, Sam Greydanus, et al. 2020) and variational integrators (Saemundsson et al. 2020; S. A. Desai, Mattheakis, and Roberts 2021) into networks. Extensions in this direction have pushed the envelope to also learn non-conservative/irreversible systems (Zhong, Dey, et al. 2020b; Yin et al. 2021; S. Desai et al. 2021; Lee et al. 2021) and contact dynamics from sparse, noisy data (Hochlehnert et al. 2021). To increase the pace of innovation, several software packages have been developed that use neural networks and the backpropagation technique of PINNs to approximate solutions of differential equations such as NeuroDiffEq (F. Chen et al. 2020), DeepXDE (L. Lu et al. 2021), and SimNet (Hennigh et al. 2020).

In spite of these developments, transfer learning remains under explored. Wang et.al (2021) show transfer learning methods can be used to stitch solutions together to resolve a large domain. Yet further work by Mattheakis et.al (2021) illustrates how a reservoir of weights can be transferred to new ICs. Here, we push these further and identify a general model-agnostic method to do one-shot inference for systems of linear ordinary and partial differential equations.

## 5.4 Method

### 5.4.1 ODEs

We define a neural network such that the approximate network solution $\psi(t)$ at time points $t$ is: $\psi(t) = H(t)_{\theta_H} W_{\theta_W} + B_{\theta_B}$. In other words, the neural network, parametrized by $\theta = [\theta_H, \theta_W, \theta_B]$, transforms the inputs $t \in \mathbb{R}^{t\times 1}$ into a high dimensional, non-linear latent space $H \in \mathbb{R}^{t\times h}$ through a composition of non-linear activations and hidden layers. Then, a linear combination of the latent space is taken, akin to reservoir computing (Jaeger et al. 2004), to obtain the solution $\psi(t)$.

To train the network, we design the final weights layer to consist of multiple outputs, i.e. $W_{\theta_W} \in \mathbb{R}^{h\times q}$. This is done so that multiple ($q$) solutions, $\psi(t) \in \mathbb{R}^{t\times q}$, can be estimated and simultaneously trained to satisfy equations that have different linear operators $D_n$ defined by different coefficients $a_i(t)$, as well as different initial conditions $\psi_{\text{ic}}$, and forces $f$. Bundle training allows us to (1) integrate the training into a single network and (2) to encourage the hidden states $H(t)$ to be versatile across equations.

At inference, the weights for the hidden layers are frozen and $H$ is computed at specific time points $\hat{t}$. The solution is therefore $\psi(\hat{t}) = H(\hat{t})W_{\text{out}}$ where $W_{\text{out}}$ is trainable. For a new set of ICs $\psi'_{\text{ic}}$, source $f'$, and differential operator $\hat{D}'_n$ the loss of the linear ODE (Eq. 5.3), becomes:

$$\mathcal{L} = \mathcal{L}_{\text{diffeq}} + \mathcal{L}_{\text{IC}} = \left(\hat{D}'_n H W_{\text{out}} - f'(t)\right)^2 + \left(\bar{D}_0 H W_{\text{out}} - \psi'_{\text{ic}}\right)^2 \tag{5.4}$$

Since Eq. 5.4 is convex, the fine-tuning of $W_{\text{out}}$ can be computed analytically. In other words, to minimize $\mathcal{L}$ we need to solve the equation $\partial\mathcal{L}/\partial W_{\text{out}} = 0$. The derivative of the first term of Eq. 5.4 is:

$$\frac{\partial \mathcal{L}_{\text{diffeq}}}{\partial W_{\text{out}}} = 2\left(\hat{D}'_n H\right)^T \left(\hat{D}'_n H W_{\text{out}} - f'(t)\right). \tag{5.5}$$

Taking the same approach for the second term of Eq. 5.4 that enforces ICs, we obtain:

$$\frac{\partial \mathcal{L}_{\text{ICs}}}{\partial W_{\text{out}}} = 2(\bar{D}_0 H)^T (\bar{D}_0 H W_{\text{out}} - \psi'_{\text{ic}}). \tag{5.6}$$

We let $\hat{D}'_n H = \hat{D}_H$ and $\bar{D}_0 H = \bar{D}_H$ to simplify the notation. Adding the loss terms together and setting them to zero yields the optimal output weights:

$$W_{\text{out}} = \left(\hat{D}_H^T \hat{D}_H + \bar{D}_H^T \bar{D}_H\right)^{-1} \left(D_H^T f'(t) + \bar{D}_H^T \psi'_{\text{ic}}\right). \tag{5.7}$$

Therefore, given any fixed hidden states $H(\hat{t})$ at fixed time-points $\hat{t}$, one can analytically compute a $W_{\text{out}}$ for any linear differential equation that minimizes 5.4. Broadly, we can think of $H$ as being a collection of non-orthogonal basis functions that can be linearly combined to determine the output function.

Note that one special outcome of this formalism is that the matrix inversion at inference is independent of the ICs $\psi'_{\text{ic}}$ and force $f'$, which means for any new ICs or $f'$, $W_{\text{out}}$ can be computed with a simple matrix multiplication if the inverse term in Eq. 5.7 is pre-computed. The benefits of this approach are multi-fold, given $H$ we achieve fast inference (order of seconds for 1000s of differential equations), eliminate the need for gradient-based optimization as no further training is required, and maintain high accuracy if $H$ is well-trained. Indeed this approach relies on determining an inverse matrix (see Appendix D.3 for details on computational complexity). If $D_H^T D_H$ has a large condition number, the matrix will have many eigenvalues close to zero - indicating ill-conditioning. Experimentally, we circumvent this issue by using regularisation or QR decomposition (see Appendix D.2).

We have shown that an analytic $W_{\text{out}}$ can only be determined for linear non-homogeneous ODEs. However, the proposed network design can still be used, as we show later, for efficient transfer learning of non-linear ODEs.

### 5.4.2 PDEs

An important outcome of the formalism for ODEs is a natural extension to linear PDEs. Many PDEs that appear in real-world problems are linear, including the diffusion equation, Laplace equation, the wave equation as well as the time-dependent Schrödinger PDE. Considering one spatial dimension $x$ and one time dimension $t$, a general linear second order differential equation takes the form:

$$\left(D^t + D^x + D^{xt} + V(t,x)\right)\psi(x,t) = f(x,t), \tag{5.8}$$

where we denote a second order time operator $D^t\psi = \sum_{i=1}^{2} a_i(t,x)\psi_t^{(i)}$, the spatial second-order operator $D^x\psi = \sum_{i=1}^{2} b_i(t,x)\psi_x^{(i)}$, and a mixed space-time operator, $D^{xt}\psi = D^{tx}\psi = c(x,t)\psi_{xt}$. The coefficients $a, b, c$ and commonly called source $f$ and potential $V$ functions, are continuous functions of $x, t$, where the lower indices indicate partial derivatives according to the notation: $\psi_\nu^{(i)} = \frac{\partial^{(i)}\psi}{\partial\nu^{(i)}}$ and $\psi_{\nu\nu'} = \frac{\partial^2\psi}{\partial\nu\nu'}$. The structure of Eq. 5.8 can generalize to higher orders and for more variables.

The last part to complete the derivation is to enforce the BCs and ICs in the loss function. For the purpose of the derivation, we use Dirichlet BCs. Thus,

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{\text{diffeq}} + \mathcal{L}_{\text{IC}} + \mathcal{L}_{\text{BCs}} \\ &= \left(\hat{D}\psi - f(t,x)\right)^2 + \left(\psi(0,x) - g(x)\right)^2 \\ &\quad + \sum_{\mu=L,R} \left(\psi(t,\mu) - B_\mu(t)\right)^2, \end{aligned} \tag{5.9}$$

where $\hat{D} = (D^t + D^x + D^{xt}) + V(t,x)$, $B_L(t)$ and $B_R(t)$ are the left and right boundary conditions, and $g(x)$ is the initial condition at $t = 0$. Similarly to the derivation for ODEs, we analytically compute $W_{\text{out}}$ of Eq. 5.9, namely we solve the equation $\partial\mathcal{L}/\partial W_{\text{out}} = 0$ considering a neural solution of the form $\psi = HW_{\text{out}}$. Starting with the first term of Eq. 5.9, we read:

$$\frac{\partial\mathcal{L}_{\text{diffeq}}}{\partial W_{\text{out}}} = 2\hat{D}_H^T(\hat{D}_H W_{\text{out}} - f(t,x)) \tag{5.10}$$

where $\hat{D}_H = \hat{D}H$. Accordingly, for the IC loss component we obtain:

$$\frac{\partial\mathcal{L}_{\text{IC}}}{\partial W_{\text{out}}} = 2H_0^T(H_0 W_{\text{out}} - g(0,x)), \tag{5.11}$$

with $H_0 = H(0,x)$. For the BCs loss components we have:

$$\frac{\partial\mathcal{L}_{\text{BCs}}}{\partial W_{\text{out}}} = \sum_{\mu=L,R} 2H_\mu^T \left(H_\mu W_{\text{out}} - B_\mu(t)\right), \tag{5.12}$$

where $H_\mu = H(t,\mu)$. Piecing this all together yields:

$$W_{\text{out}} = \left(\hat{D}_H^T\hat{D}_H + \sum_{\mu=0,L,R} H_\mu^T H_\mu\right)^{-1} \left(\hat{D}_H^T f(t,x) + \sum_{\mu=0,L,R} H_\mu^T Q_\mu(t,x)\right), \tag{5.13}$$

where $Q_0 = g(x)$, $Q_L = B_L(t)$, and $Q_R = B_R(t)$.

We therefore show it is equally feasible to obtain an analytic set of linear weights to determine solutions to PDEs. Indeed, the accuracy of the solution depends heavily on how well the hidden states $H$ span the solution space. To encourage a representative hidden space, we typically bundle train a single network on different equations, namely a network with multiple outputs. We find that by using more initial and boundary conditions at training, the generalization performance improves (see. Appendix D.4). That is to say, the hidden states are improved by batch training on multiple conditions simultaneously. To do this, we design a new type of PINN with multiple output nodes for which each node represents a solution to the same differential equation under different initial conditions. Doing this allows us to train a PINN to satisfy multiple initial conditions simultaneously using the same latent basis. It must be noted that while the work in this chapter focused on transfer learning to new initial and boundary conditions of the same differential equation, it is certainly plausible to do this for entirely different equations as long as they have the same inputs. A model of this would consist of a PINN with multiple output nodes, each of which would be used to approximate the solution of a different differential equation in training. In other words, each output node would satisfy a different differential equation using the same hidden layers. This training regime would make the hidden layers of the network robust to different equations as well as different initial and boundary conditions. Of course, this will lead to problems with the F-principle if the solution spaces are vastly different (Xu et al. 2019). We leave this for future exploration.

## 5.5 Results

We investigate our method on numerous well known differential equations of practical interest. We present a summary of our results in table 5.1. The full training regime, network design and test conditions are as follows with more details regarding the systems under investigation being provided later in the chapter where results are presented.

| Differential Equation | # Training Bundles | # Test Bundles | Test Time (s) | Test Accuracy (MSE) |
|---|---|---|---|---|
| First-order linear ODEs | 10 | 1000 | $7.4 \times 10^{-3}$ | $1.35 \pm 1.65 \times 10^{-10}$ |
| Second-order linear ODEs | 10 | 1000 | $3.4 \times 10^{-3}$ | $2.84 \pm 1.87 \times 10^{-9}$ |
| Coupled linear oscillators | 10 | 100 | $4.7 \times 10^{-2}$ | $2.29 \pm 4.74 \times 10^{-12}$ |
| Nonlinear oscillator | 5 | 30 | 5.2 | $1.47 \pm 3.88 \times 10^{-4}$ |
| Poisson | 4 | 100 | 33.2 | $3.60 \pm 8.84 \times 10^{-5}$ |
| Schrödinger | 3 | 400 | 19.4 | $5.02 \pm 8.92 \times 10^{-5}$ |

**Table 5.1:** Summary results of our method on all the systems investigated. Training on a few bundles is sufficient to rapidly and accurately scale to many unseen conditions. Note that the nonlinear oscillator is optimized using gradient descent whereas the other methods are all optimized using analytic $W_{\text{out}}$. For reference, training a PINN requires several thousand iterations to obtain accurate solutions, where a single iteration costs 0.07s. All times are reported for a CPU.

## Training Configuration

All models are trained using an Adam optimizer with a learning rate of $10^{-3}$. The networks are all trained on a Macbook Pro, 2.2 GHz Intel Core i7, 16 Gb RAM. We use 64-bit tensors. The network configurations and training collocation points can be found in table 5.2.

| Differential Equation | Architecture (N bundles) | # training Bundles | Training Iterations | # Training Collocation Points | Evaluation Domain | Evaluation Deltas | Activations |
|---|---|---|---|---|---|---|---|
| First-Order Linear Inhomogeneous | 1-100-100-1*N | 10 | 10000 | 30 | t in [0,3] | dt = 0.1 | tanh |
| Second-Order Linear Inhomogeneous | 1-100-100-1*N | 10 | 10000 | 30 | t in [0,3] | dt = 0.05 | tanh |
| Coupled-Oscillator | 1-100-100-2*N | 10 | 10000 | 50 | t in [0,10] | dt = 0.01 | sin |
| Non-Linear Oscillator | 1-100-100-1*N | 5 | 10000 | 60 | t in [0,3] | dt = 0.05 | sin |
| Poisson | 2-100-100-1*N | 4 | 40000 | 1000 | x in [0,1], t in [0,1] | dx = 0.01, dt = 0.01 | sin |
| Schroedinger | 2-100-100-2*N | 3 | 40000 | 1000 | x in [-10,10], t in [0,1] | dx = 0.1, dt = 0.01 | $\alpha \sin + (1-\alpha)\tanh$ |

**Table 5.2:** Network configurations for each of the systems investigated.

Here we briefly outline each of the differential equations we investigate and the parameter sets we select from for training.

**First-Order Linear Inhomogeneous Systems**

The equation is of the form:

$$\dot{u} + a(t)u = f(t), \tag{5.14}$$

subjected to an initial condition $u_0$. We sample within:

$$f \in \{\cos(t), \sin(t), t\},$$
$$a \in \{t, t^2, 1\},$$
$$(u_0) \in [-5, 5].$$

**Second-Order Linear Inhomogeneous Systems**

The equation is of the form:

$$\ddot{u} + a_1(t)\dot{u} + a(t)u = f(t) \tag{5.15}$$

with initial conditions $u_0$ and $\dot{u}_0$. We sample within:

$$\begin{aligned} f &\in \{1, t, \cos(t), \sin(t)\}, \\ a &\in \{1, 3t, t^2\}, \\ a_1 &\in \{1, t^2, t^3\}, \\ (u_0, \dot{u}_0) &\in [-5, 5]. \end{aligned}$$

**Coupled Oscillator System**

The equation is of the form:

$$\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} \begin{bmatrix} \ddot{u}_1 \\ \ddot{u}_2 \end{bmatrix} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_1 + k_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{5.16}$$

We sample:

$$\begin{aligned} m &\in [1, 2], \\ (k_1, k_2) &\in [0.5, 4.5], \\ (u_0, u_1, \dot{u}_{1,0}, \dot{u}_{2,0}) &\in [-1.5, 1.5]. \end{aligned}$$

**Nonlinear Oscillator System**

The equation is:

$$\ddot{u} + u + u^3 = 0. \tag{5.17}$$

We sample within:

$$(u_0, \dot{u}_0) \in [0.5, 2].$$

**Poisson equation**

$$\nabla^2 u = \rho \tag{5.18}$$

we sample:

$$\rho \in \{\sin(x)\sin(y), \sin(2x)\sin(2y), \sin(3x)\sin(3y), \sin(4x)\sin(4y)\}.$$

**Schrödinger System**

$$\begin{bmatrix} \dot{u}_R \\ \dot{u}_I \end{bmatrix} = \begin{bmatrix} 0 & -\hbar/2m \\ \hbar/2m & 0 \end{bmatrix} \begin{bmatrix} u''_R \\ u''_I \end{bmatrix} \tag{5.19}$$

where $\psi = (u_R, u_I)$ and:

$$\psi(x,0) = \frac{1}{\pi^{1/4}\sqrt{\sigma}} e^{-(x-x_0)^2/(2\sigma^2)+ip_0x/\hbar}. \tag{5.20}$$

and

$$\sigma \in \{0.5, 0.6, 0.7\},$$
$$p_0 \in \{1, 2, 3\}.$$

For ODEs we report accuracy as the MSE of the residual: $|\hat{D}_n\phi_\theta(t) - f(t)|^2$. For PDEs we report accuracy as the MSE between the predicted solution and the analytic solution: $|\psi_{gt} - \psi_{pred}|^2$. In addition, we highlight how our solver can be used to tackle specific challenges with learning from these equations. Code is made available here [1].

### 5.5.1 ODEs

**Linear ODEs**

To test the performance of our approach, we train both first- and second-order methods of linear non-homogeneous differential equations. For first order ODEs, the equation is defined by the operator of Eq. 5.2 with $n = 1$ and by specifying three quantities: the time-dependent coefficients $a_0$, the forces $f$, and the ICs. As such, any first-order linear non-homogeneous ODE can be defined by the tuple $(a_0, f, \text{ICs})$. Given a pre-defined list of options for each quantity in the tuples listed above, we randomly sample 10 tuples for training. We batch train our model on all the equations simultaneously (i.e. $W_{\text{out}} \in \mathbb{R}^{t\times 10}$). We then carry out inference on

[1] https://github.com/shaandesai1/TransferDE

1000 randomly sampled test tuples using analytic $W_{\text{out}}$ from Eq. 5.7. Results are presented in table 5.1. It is clear that for first-order differential equations, transfer learning analytic $W_{\text{out}}$ is significantly advantageous since we obtain high-fidelity solutions. As such, we take a similar approach for second order differential equations described by the operator of Eq. 5.2 for $n = 2$. We plot the results of 20 ODEs from the test set and compute their residuals in Fig. 5.1 where $\dot{\psi} = d\psi/dt$. Indeed, the overall test accuracy depends on how well the hidden states span the space of differential equations. We typically find that more training bundles results in better test performance (see Appendix D.4).

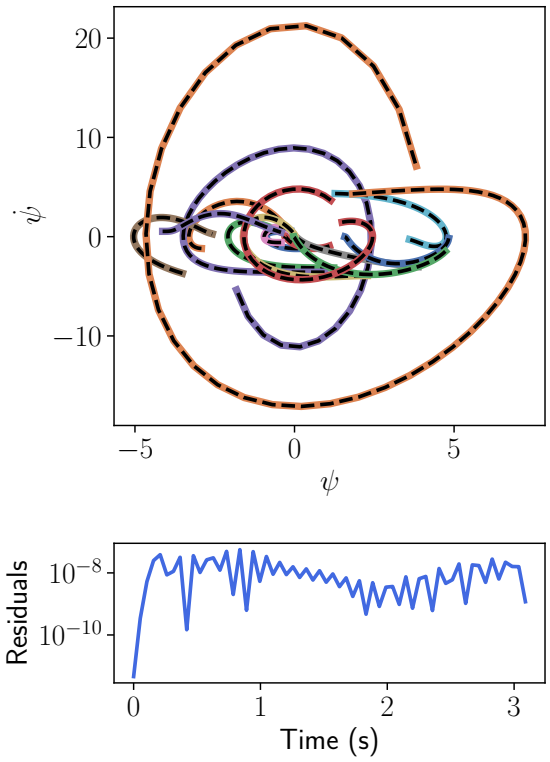



**Figure 5.1:** Predicted (colored) versus ground truth (dashed black) phase space, namely a plot of space against velocity for different times, to 20 second-order non-homogeneous ordinary differential equations. Average residuals are shown in the bottom panel.

### Systems of ODEs

Since an analytic $W_{\text{out}}$ can be computed for linear ODEs, the method naturally extends to systems of linear differential equations (see Appendix D.1 for derivation). To highlight this, we investigate a system of linear second-order ODEs of the form

$\ddot{\boldsymbol{\psi}} = A\boldsymbol{\psi}$ that describe a system of two coupled oscillators where $\boldsymbol{\psi} = [\psi_1, \psi_2]$ and $A$ describes the coupling. The equation is of the form:

$$\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} \begin{bmatrix} \ddot{\psi}_1 \\ \ddot{\psi}_2 \end{bmatrix} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_1 + k_2 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}. \tag{5.21}$$

We train the network to satisfy 10 different $\{m, k_1, k_2\}$ values and initial conditions $[\boldsymbol{\psi}_0, \dot{\boldsymbol{\psi}}_0]$ as is defined above such that the pre-trained network can be used to instantaneously compute accurate solutions for different ICs of the coupled masses. We report the result of testing 100 different systems sampled from the same range as training in table 5.1. Furthermore, we investigate an interesting application in which the network can be exploited to identify initial conditions of a coupled-oscillator system capable of inducing *beats* - when two normal mode frequencies come close (see Fig. 5.2) (Schwartz 2017).

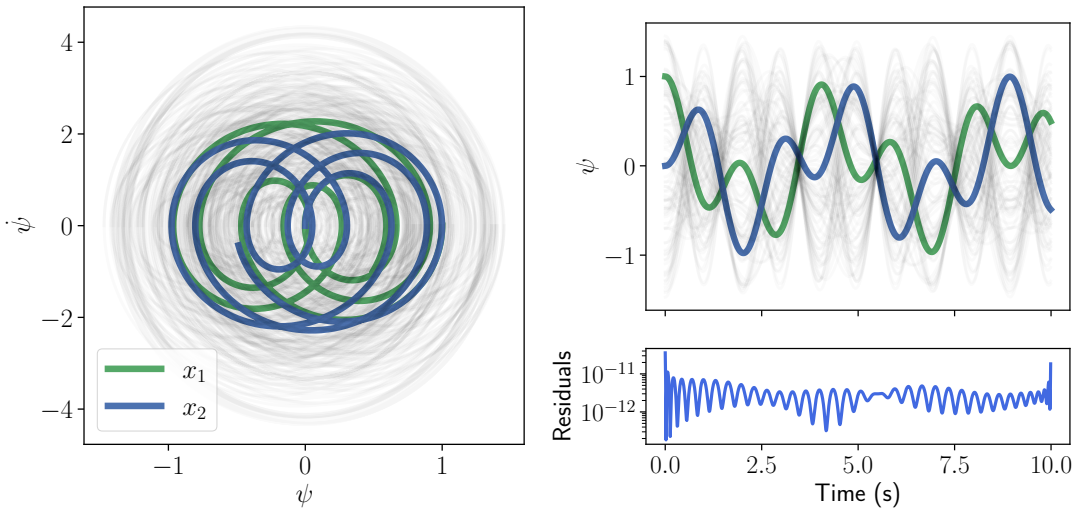



**Figure 5.2:** Phase space trajectories of the coupled oscillator system for fixed mass and spring constants (top) and spatial solutions (middle). One solution that induces beats is highlighted in color while the other solutions appear in grey. The average residuals of the total realizations are shown in the bottom panel. The initial state of the masses influences how close the normal mode frequencies get. Our network can identify solutions to all 100 initial conditions in $\sim 10^{-2}$ seconds.

### Nonlinear ODE

Until this point, we have only investigated the success of fine-tuning $W_{\text{out}}$ analytically for linear ODEs, nevertheless our network proposal can still be exploited to transfer

learn nonlinear ODEs. To do so, we replace the computation of analytic $W_{\text{out}}$ with gradient-based optimization. Note, since the hidden weights are frozen, the final hidden activations can be pre-computed given sampling points $t$ for efficient optimization. We use this approach to solve a Hamiltonian nonlinear system described by the ODE:

$$\ddot{\psi} = -\psi - \psi^3, \tag{5.22}$$

that conserves energy given by the Hamiltonian:

$$\mathcal{H} = \frac{\dot{\psi}^2}{2} + \frac{\psi^2}{2} + \frac{\psi^4}{4}. \tag{5.23}$$

We train the system on 5 initial positions $\psi_0$ randomly sampled in the range [0.5, 2.0] and with initial velocity $\dot{\psi}_0 = 0$. The loss function during training consists of (1) a differential equation loss, (2) an initial condition loss, and (3) an energy conservation loss penalty. The energy loss enforces the Hamiltonian at all points in time to be the same, namely $\mathcal{L}_{\text{E}} = (\mathcal{H}(\psi, \dot{\psi}) - \mathcal{H}(\psi_0, \dot{\psi}_0))^2$ (Mattheakis, Sondak, et al. 2020). We then evaluate the performance of the hidden states on 30 ICs sampled in the same range (see Fig. 5.3). Since we freeze the hidden layers, we can pre-compute the hidden activations $H(\bar{t})$ at fixed time $\bar{t}$ and then fine-tune $W_{\text{out}}$ using gradient descent for 5000 epochs. Note that the optimization can be done using other methods as well, including L-BFGS since the entire problem is reduced to convex optimization.

### 5.5.2 PDEs

PDEs can be used to model complex spatio-temporal systems,s making them of practical interest in numerous domains. Typically, the most well-studied PDEs are linear and include the diffusion, Poisson, and wave equations. To benchmark the performance of this approach, we investigate the Poisson equation and the time-dependent Schrödinger equation.

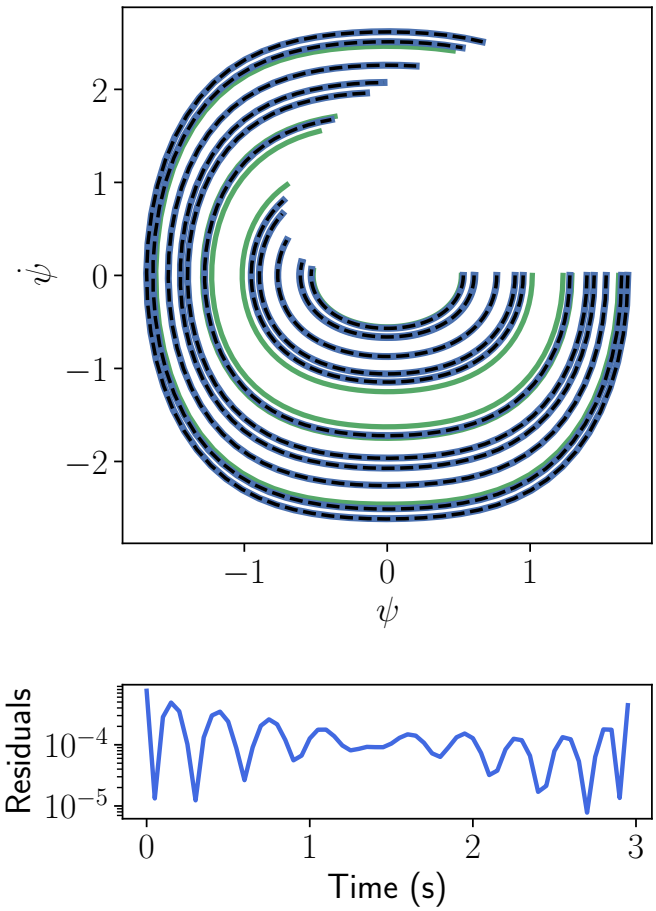



**Figure 5.3:** Top: phase space of predicted trajectories of a nonlinear oscillator system. The training curves are shown in green and the test in blue. Dashed black lines represent ground truth solutions. Bottom: average residuals of 30 predicted solutions across different initial conditions.

### Poisson Equation

The Poisson equation is an extensively studied PDE in physics, typically used to identify an electrostatic potential $\psi$ given a charge distribution $\rho$. In 2-D, it can be described by:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \rho(x, y). \tag{5.24}$$

We define this PDEs in the domain $x \in [x_L, x_R]$ and $y \in [y_B, y_T]$ with BCs:

$$\psi(x_L, y) = \psi(x_R, y) = \psi(x, y_B) = \psi(x, y_T) = 0. \tag{5.25}$$

We train the network on 4 different charge distributions $\rho(x, y) = \sin(k\pi x)\sin(k\pi y)$ for $k \in 1, 2, 3, 4$. We then evaluate the performance of our network in two settings.

The first is an ablation across 100 linearly spaced values of $k$ in $[1, 4]$ (see table 5.1). As a second experiment we test the proposed transfer learning method on a harder testing force function of the form:

$$\rho_{\text{test}} = \frac{1}{4}\sum_{k=1}^{4}(-1)^{k+1}2k\sin(k\pi x)\sin(k\pi y). \tag{5.26}$$

The solution is shown in the top graph of Fig. 5.4. To assess the network performance, we plot in the lower graph of Fig. 5.4 the mean square error (MSE) computed between the predicted $\psi(x, y)$ and the analytical solution which reads:

$$\psi(x, y) = \frac{-1}{2(k\pi)^2}\sin(k\pi x)\sin(k\pi y). \tag{5.27}$$

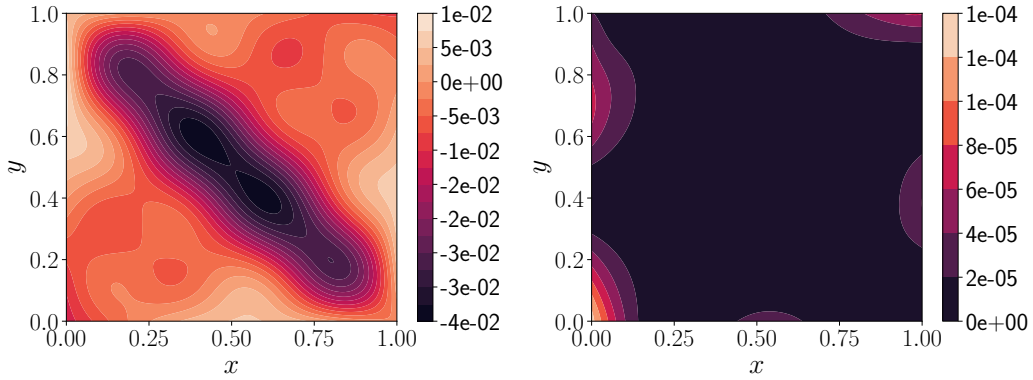



**Figure 5.4:** Predicted solution (top) of the Poisson equation with an initial charge distribution $\rho(x, y)$ composed of multiple frequencies $k$. The network is pre-trained on the individual frequencies and can obtain the solution to the combination in one-shot (35s) with high fidelity/low MSE (bottom).

### Time-Dependent Schrödinger Equation

In quantum mechanics the time-dependent Schrödinger equation describes the propagation of a wavefunction through space and time. The PDE in one-dimensional space is of the form:

$$i\hbar\frac{\partial}{\partial t}\psi(x, t) = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x)\right]\psi(x, t), \tag{5.28}$$

where $\psi(x, t)$ is a complex-valued function called wave-function, $V(x)$ is a stationary potential function, $m$ is the mass, and $\hbar$ is a constant. We investigate the quantum free-particle evolution for which $V(x) = 0$ for several initial states $\psi(x, 0)$.

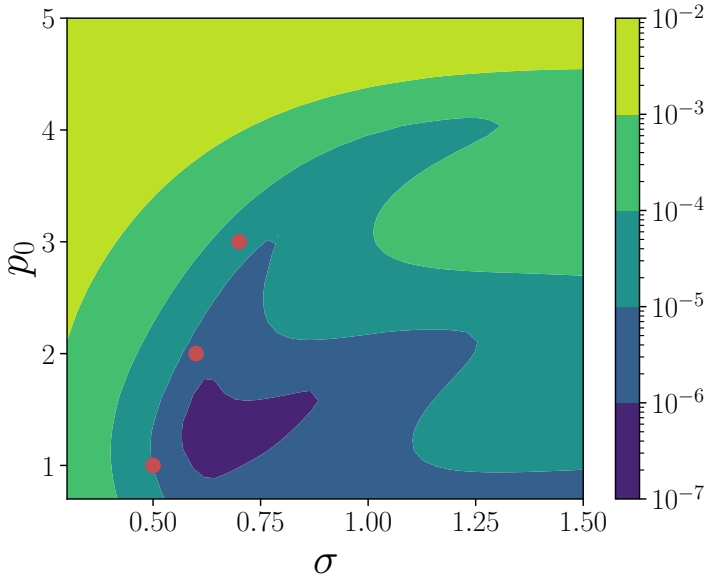



**Figure 5.5:** MSE between predicted and analytic solutions $|\psi|^2$ as a function of $\sigma$ and $p_0$. Red circles represent the three configurations for which the network was batch trained. We see that as $p_0$ increases, transfer learning $W_{\text{out}}$ becomes less effective because of the F-principle bottleneck for PINNs.

To train the complex-valued wave-function, we separate the real $\psi_R$ and imaginary $\psi_I$ parts (M. Raissi et al. 2019), namely $\psi(x,t) = \psi_R(x,t) + i\psi_I(x,t)$. By plugging the above form of $\psi$ into Eq. 5.28 we obtain a coupled system of real-valued PDEs as:

$$\frac{\partial}{\partial t}\begin{bmatrix}\psi_R \\ \psi_I\end{bmatrix} = \begin{bmatrix}0 & -\hbar/2m \\ \hbar/2m & 0\end{bmatrix}\frac{\partial^2}{\partial x^2}\begin{bmatrix}\psi_R \\ \psi_I\end{bmatrix}, \tag{5.29}$$

which is of the form $\boldsymbol{\psi}_t = A\boldsymbol{\psi}_{xx}$. The system is linear and thus, we can obtain analytic $W_{\text{out}}$. We consider a network with two outputs per equation associating with $\psi_R$ and $\psi_I$, where each output is, respectively, described by a set of weights as $W_{\text{out}} = [W_{\text{R}}, W_{\text{I}}]^T$. Then, the network solutions read:

$$\begin{bmatrix}\psi_R \\ \psi_I\end{bmatrix} = \begin{bmatrix}H & 0 \\ 0 & H\end{bmatrix}\begin{bmatrix}W_R \\ W_I\end{bmatrix}, \tag{5.30}$$

By taking the $L_2$ loss of Eq. 5.29 as well as the BCs and ICs we obtain:

$$W_{\text{out}} = (D_H^T D_H + H_0^T H_0 + H_d^T H_d + \dot{H}_d^T \dot{H}_d)^{-1}(H_0^T \psi_0), \tag{5.31}$$

where $H_0 = H(0,x)$, $H_d = H(t,L) - H(t,R)$, and $\dot{H}_d = H_x(t,L) - H_x(t,R)$.

To investigate a particular set of solutions, we define the initial condition for this problem as:

$$\psi(x,0) = \frac{1}{\pi^{1/4}\sqrt{\sigma}} e^{-(x-x_0)^2/(2\sigma^2)+ip_0x/\hbar}, \tag{5.32}$$

that leads to the exact solution:

$$\psi(x,t) = \frac{e^{-\frac{(x-(x_0+p_0t/m))^2}{2\sigma^2(1+i\hbar t/m\sigma^2)}} e^{i(p_0x-Et)/\hbar}}{\pi^{1/4}\sqrt{\sigma(1+i\hbar t/m\sigma^2)}}, \tag{5.33}$$

where $E = p_0^2/2m$.

We train a network simultaneously on three solutions of Eq. (5.28) with pairs of $\sigma, p_0$ such that the network is trained for $(\sigma, p_0) = \{(0.5,1),(0.6,2),(0.6,3)\}$. We show that by using only 3 training ICs, accurate solutions to multiple other configurations can be obtained instantly and with high accuracy (see Fig. 5.5). We measure and present in Fig. 5.5 the accuracy of our predicted solution by computing the MSE across time and space against the analytic solution Eq. 5.33. Furthermore, we show that near the bundles, the predicted real-imaginary complex space under different $\sigma, p_0$ pairs tightly couples to the analytic solution (see Fig. 5.6). Our results also highlight the F-principle, a conclusion drawn about deep networks which shows that high frequency components require more training (Xu et al. 2019) as can be seen in Fig. 5.5 by looking at higher $p_0$ values which induce higher frequency components in the solution $\psi$.

## 5.6 Conclusion

We have extensively shown how PINNs can be batch trained on a family of differential equations to learn a rich latent space that can be exploited for transfer learning. For linear systems of ODEs and PDEs, the transfer can be reduced to computing a closed-form solution for $W_{\text{out}}$ resulting in one-shot inference. This analytic solution significantly speeds up inference on unseen differential equations by orders of magnitude, and can therefore replace or augment traditional transfer learning. In particular, we show that such a network can identify first-order ODEs, second-order

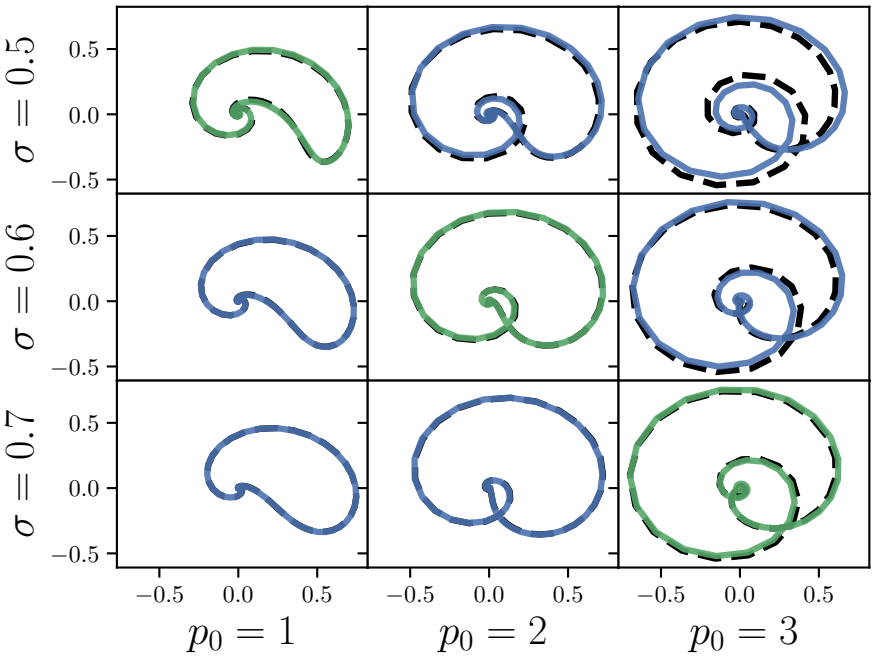



**Figure 5.6:** Real (x-axis) against imaginary (y-axis) wave-functions of the predicted solutions $\psi(T_{\max}, x)$ for different realizations of $\sigma, p_0$ with solid and dashed lines representing the predicted and ground truth solutions. The diagonal configurations (green) are used for training. Non-diagonals (blue) constitute test configurations.

coupled ODEs, Poisson and Schrödinger equations with high levels of accuracy within a few seconds. Furthermore, in the nonlinear setting, where a closed-form analytic $W_{\text{out}}$ is not derived, we show that our approach can still be used with gradient descent to identify accurate solutions to the nonlinear oscillator system. These results are particularly important to practitioners who seek to rapidly identify accurate solutions to differential equations of the same type, namely of the same order, but under different conditions and coefficients. Indeed many new applications may arise as a consequence of this approach, from transfer learning on large domains, to solving high dimensional linear PDEs. Future work in this direction may adapt to data-dependent settings, incorporate non-linear outputs to develop one-shot training for nonlinear equations, and investigate properties of the learnt hidden space.

*So we beat on, boats against the current*
*borne back ceaselessly into the past.*

— F. Scott Fitzgerald

# 6 Conclusion and Future Work

## Contents



## 6.1 Conclusion

Our ability to understand the physical world hinges tightly on developing mathematical and computational models to describe physical phenomena. While traditional numerical approaches, such as finite element and Runge-Kutta methods, have been extensively developed, they continue to face numerous computational limitations. In recent years it has been extensively shown that modern machine learning techniques, such as neural networks, provide a better pathway to modeling complex relationships from multi-modal physical data - data typically acquired from multiple sources and sensors. In particular, many studies (Karniadakis et al. 2021; Battaglia, Pascanu, et al. 2016; Samuel Greydanus et al. 2019) show that endowing these models with partial prior knowledge of the physical system at hand can significantly accelerate

training as well as improve inference, data-efficiency and interpretability. In this thesis we explored such learning biases extensively across a range of well-known systems. At a higher level we find that inductive biases, when carefully selected, can significantly improve learning from sparse, noisy data. This conclusion is drawn from our investigation of learning biases in three domains, (1) for energy-conserving systems, (2) for damped and driven systems, and (3) for general transfer learning of solutions identified by physics-informed neural networks.

In Chapter 3, aimed at improving energy conserving systems, we investigated numerous recent innovations capable of learning and forecasting the dynamics of energy-conserving systems using inductive biases in neural networks. By deconstructing and grouping these recent innovations into individual inductive bias segments, we were able to conduct an extensive ablation across all combinations of the individual biases. This procedure not only generalized existing approaches but also helped identify a new class of networks we termed Variational Integrator Graph Networks (VIGNs). We demonstrated that VIGNs are consistently the most performant across a host of application domains because they bring symplecticity, energy and structure constraints into a single network. As a consequence of conducting the ablation, our results also highlight which biases help in particular settings. For example, we find that symplectic constraints are deeply advantageous to modeling chaotic systems, graph networks are fundamental to learning many-body systems and that energy constraints, which leverage existing knowledge of the underlying system, are consistently more effective. Most importantly, the ablation helps us identify how important individual learning biases are within combinations, a study that was missing in the literature.

Many real-world dynamical systems do not conserve energy as they undergo time-dependent forcing and damping. In Chapter 4 we introduced a port-Hamiltonian formalism into neural networks to tackle the challenge of learning forced and damped systems alike from data. We showcased how such a network is capable of enforcing a known Hamiltonian constraint while also learning a time-dependent force and damping term essential for dynamic forecasting as well as system identification.

The method is tested on a number of systems and consistently outperforms other approaches at learning from such data. A special outcome of this study is the ability of our network to effectively recover the Poincaré section of a chaotic driven system. Unlike existing literature, one of the major advances of this work is the assumption that the control or forcing signal is not known a priori, making the forecasting problem harder. Needless to say, an appropriately chosen port-Hamiltonian embedded into a neural network is able to learn the necessary components directly from sparse, noisy simulated data.

Inherent to any dynamical system is an underlying differential equation. Recently, it was shown that when a differential equation is known a priori, physics-informed neural networks (Maziar Raissi et al. 2017) can be used to identify their solutions. In Chapter 5, we investigated how a PINN, simultaneously pre-trained on a number of differential equations, can be exploited for transfer learning. In particular, we showed that for linear systems of differential equations, the transfer can be done in one-shot without retraining an entire network. This transfer technique is able to preserve high-fidelity solutions on thousands of differential equations making it a powerful paradigm for solving high-order partial differential equations under many conditions. We showed this approach on a number of linear systems such as first and second-order ODEs, as well as the Poisson and Schrödinger equations. Furthermore, we prove the technique is also able to provide effective solutions to non-linear oscillators. This is the first approach to truly show how important transfer learning is for PINNs and will certainly open many doors in scientific machine learning to expedite the learning of differential equations with low computational cost.

In all these studies, we consistently found that physics-informed priors embedded in neural networks are crucial to making neural networks data efficient and accurate at inferring and forecasting the dynamics of complex physical phenomena from data. In addition to generalization performance, these models significantly improve interpretability. In VIGN, the solution is interpretable because each $[q, p]$ pair can be mapped to an energy due to Hamiltonian constraints. This means that the network output has a physical meaning. In PHNN, the network can map inputs

$[q, p]$ across time to uncover the underlying force, damping and energy meaning the underlying form of the system can be uncovered. In our transfer work we show that PINNs can be used to interpret solution spaces to differential equations via the learnt hidden states. These advances open many doors for further investigation.

## 6.2 Future Work

### 6.2.1 Physics-Informed Networks

Despite progress in simulating multi-physics problems using numerical discretization of partial differential equations (PDEs), one still cannot seamlessly incorporate noisy data into standard algorithms, mesh generation remains complex and high-dimensional problems governed by parameterized PDEs remain difficult to solve due to the curse of dimensionality. These limitations make PINNs an exciting new direction to explore as they overcome some of these challenges. Despite this, PINNs come with their own limitations. They are not inherently causal, since the network treats each point as an independent input even though adjacent points may be correlated - though some very recent work shows how this might be tackled to drastically improve model performance (S. Wang, Sankaran, et al. 2022). PINNs also require special optimization routines that leverage multiple optimizers in sequence such as Adam and LGBM, to achieve accurate results which not only increases training time but also complicates the optimization procedure. Standard PINNs also don't take into account uncertainty. These limitations create natural pathways for future methodology advances. An important, yet open, challenge for PINNs is their deployment in real-world applications. Lately, there has been a surge of interest in using PINNs to model complex multiphysics systems such as turbulent flow. These flows can have complex frequency domain components which appear extremely difficult to learn using PINNs due to the 'F-principle' (Xu et al. 2019). The F-principle is a proof that shows that deep networks, including PINNs, are biased towards identifying low frequency solutions early in training and then gradually build up high frequency components over long training horizons. In other words, rapidly training a PINN to learn solutions with high frequency

components needs to be addressed in order to make PINNs a viable solution for modelling complex physical phenomena. Given all of this, a number of limitations need to be overcome to make PINNs viable.

### 6.2.2 Applications

The need to approximate the dynamics of physical systems from sparse, noisy data is highly relevant in a host of domains. Here, we detail a few examples of how the networks presented in this thesis can be used to address some real world applications:

VIGN is a neural network capable of forecasting the dynamics of energy conserving trajectories in which noisy data is available but the underlying ordinary differential equation is not known. One application area is in the motion of astronomical objects. Astrophysicists are constantly trying to understand the dynamics of celestial bodies. Doing so allows them to uncover fundamental relationships that exist between these bodies and develop fundamental theories regarding their evolution, for example the work of (Lemos et al. 2022). VIGN is perfectly aligned to uncover these dynamics from the typically sparse, noisy data found in astronomy, thus eliminating the need to develop complex equations designed to approximate this behaviour. Further applications lie in interacting particle systems. Spring-particle models are used extensively in materials modelling and design. For example, such as the time evolution of a 2D material under an initial force. We can consider an example in which the material is microscopic and fragile and collecting samples can be difficult. As such, having a system that can learn a spring-particle model from the materials data, either generated through complex, expensive models or captured from real-world experiments can significantly speed-up the ability to prototype these materials over long time horizons with a few data-points. This application falls under a larger umbrella of surrogate models - models where an efficient network approximation is used in tandem with an expensive computational model to evolve the dynamics of a physical system

The Port-Hamiltonian method developed here goes beyond energy-conserving systems, uncovering and forecasting the dynamics of damped and driven systems

from sparse noisy data. They are particularly well-suited for modelling mechatronic motion and inferring their dynamics. For example, robot motion can often be characterized using affine control. In many cases, the control signal is known a priori and used in tandem with sparse data to uncover the dynamics of a physical system over time. However, while the input control signal may be known, the external forces that influence the robot may not be easy to model. As such, using the PHNN is ideal, as prior knowledge of the force is not needed and can be uncovered. Importantly, by design, the network can separate the force from the damping and the Hamiltonian thus creating a more interpretable model. In addition, a number of engineering applications employ a Port-Hamiltonian formalism to model the dynamics of charge-discharge cycles in circuits. Using the PHNN to uncover such dynamics from data allows engineers to rapidly identify the energy sources and sinks. This is likely to have high impact in large, complex circuits with the PHNN aiding as a diagnostic tool. Indeed machine learning techniques are already being applied to batteries (Aitio et al. 2021; W. Li et al. 2021) illustrating the potential for PHNN type models.

PINNs are excellent at solving known equations and embedding them with data - making them ideal for inverse problems. Importantly, the ability to rapidly prototype multiple solutions to an equation can be achieved using Transfer-PINNs. One application example is in epidemic modeling. Many epidemic models are designed as compartment models governed by known differential equations for which parameters need to be estimated. Given data of an epidemic and a well-known compartmental model capable of describing such data, a PINN can be used to learn the parameters. For such models, it is typically necessary to understand what outcomes look like under new parameters that might be perturbations of the learnt parameters - these perturbations might reflect different governmental interventions such as lockdowns. Our work shows that it is possible to identify 1000s of solutions under 1000s of perturbations (policies) rapidly and with high fidelity. Doing so would allow us to identify the best policy to use for intervention. There is already a great deal of existing literature that shows how compartment

models, described by differential equations can be solved using machine learning and PINNs (Kaxiras et al. 2020; Grimm et al. 2021) making this an interesting and important application given the recent Covid-19 pandemic.

Although we extensively study physical systems and priors linked to such systems, the concept of embedding priors extends far beyond physics. In particular, physics-informed neural networks are capable of solving general ODEs, PDEs and stochastic differential equations. As such, this would suggest that the physics-informed perspective can be broadened to a more general system-informed perspective as long as priors linked to the system can be embedded into the ML pipeline. One such example is in a socioeconomic setting. The socioeconomics of a region can be described by a stochastic differential equation over a graph for which nodes can be considered specific sub-regions. In this setting, it is possible to embed priors linked to specific sub-regions such as a neighbourhood's income flow, population flow and municipal interventions to improve predictive performance.

### 6.2.3 Data Challenges

Collecting real-world data from physical systems is a challenging process, subject to noise and sparsity. To capture these limitations in our simulated data, we generated sparse data points ($N \sim 1000$) with a noise-to-signal ratio of 20% in most settings. Specifically, data points are typically generated with high order (8th) integrators and small $\Delta t$ time steps between points. Doing so guarantees the generated points have minimal error due to numerical integration. These finer sampled points, $\Delta t = 0.001$, are then sub-sampled, $\Delta t = 0.1$, to account for sparsity and to better capture the data collection process in the real world. Note that the sub-sampled points ensure the timescale is associated with the phenomena of interest. In addition, noise, sampled from a Gaussian distribution, is added to the sub-sampled points which are used to train our networks.

Indeed the design of our networks depend heavily on the available data. Firstly, the $\Delta t$ timestep used in data generation can significantly impact model performance for neuralODE-like networks that are susceptible to both network approximation

error and numerical integration error due to their embedded integrators. From our experiments, we see that increasing the order of the embedded integrator dramatically increases both training and test performance, as doing so reduces the numerical integration error and allows the network to reduce its approximation error. Furthermore, the variance of the added noise also impacts performance. Quite evidently, as the noise-to-signal ratio increases, models tend to perform poorly as can be seen in the ablation of pHNN. Importantly, the addition of inductive biases improves the data-efficiency of our models as is corroborated by other work (see (Zhong, Dey, et al. 2020a; Saemundsson et al. 2020)).

These findings highlight the importance of ensuring our models are robust to uncertainty in our data as well as being able to inform us of their statistical guarantees. To this end, future work should investigate how inductive biases in networks might handle uncertainty in input data but also overcome the numerical limitations of embedded integrators, for example, through the use of modern deep learning techniques such as transformers (Geneva et al. 2021).

### 6.2.4 Model Extensions

While the models presented in this work highlight significant improvements over existing baselines, additional work is required to improve these models.

VIGN uses a recurrent neural network similar to NeuralODE to embed symplectic integrators. For short training horizons, this approach is perfectly reasonable. However, for long integration horizons, the vanishing gradient problem needs to be addressed. While NeuralODE works well with runge-kutta methods, there is no existing literature that highlights how neuralODE can be extended to symplectic integrators. As such, it would be highly advantageous to develop such a method in order to have a memory efficient, long training horizon VIGN approach that is likely to be more stable, accurate and efficient.

PHNN makes assumptions about the training data. As such, further generalizations can be made to the framework to capture more of the underlying dynamics, for example, decomposing the forcing function into a spatial and temporal component.

In addition, the optimization of the L1 penalties is done using a grid search. It is possible to use bayesian optimization approaches which could significantly improve parameter identification in such settings.

While in both VIGN and PHNN a form of integrative recurrent neural network is employed, an emergent successful alternative is transformers. It has been shown that transformers can be recast as Adam based integrators that have more expressive power as they capture derivatives of the previous states to approximate the current state (B. Li et al. 2021). This could be a powerful technique to extend both VIGN and PHNN in solving chaotic trajectories and improving accuracy.

These proposals are by no means exhaustive, but certainly indicate how future work in these directions might progress to tackle harder, application-specific problems. In this thesis, we have shown that learning biases bring many advantages to neural networks, which when carefully designed can lead to better inference as well as the ability to transfer solutions. As physics and machine learning continue to progress, such hybrid models will be at the heart of scientific progress.

# Appendices

# A

# Numerical Integrators

## A.1 Introduction

In Chapter 2, section 2.1.1, we introduced the notion of approximating an integral with a numerical method. We argued that using such an approach to approximate the definite integral:

$$\int_a^b f(x)\mathrm{d}x, \tag{A.1}$$

is necessary in many situations for which the integral is not analytically tractable. Numerical integrators are algorithms designed to approximate the integral above by discretizing it and evaluating the function $f(x)$ at specific points. Furthermore, the choice of evaluations, when carefully selected, influences the degree of accuracy. One well known numerical integrator is the midpoint method in which:

$$\int_a^b f(x)\mathrm{d}x = (b-a)\frac{f(a+b)}{2}. \tag{A.2}$$

Here, a single evaluation of the function at the midpoint of the integration domain is made. The error, typically reported as a function of the discrete step-size, $h$, for this method is $\mathcal{O}(h^2)$. To obtain higher-orders of accuracy, practitioners typically refer to Runge-Kutta integrators (Runge 1895).

## A.2 Runge-Kutta Integrators

Runge-Kutta integrators are a family of implicit and explicit iterative methods used to approximate solutions to ODEs. The integrators typically require numerous points of evaluation of the function $f(x)$ combined with specific coefficients. The coefficients can be summarized in a Butcher Tableau (Butcher 2008) such as Table. A.1.

| | | | |
|---|---|---|---|
| $c_1$ | $a_{11}$ | $\dots$ | $a_{1s}$ |
| $\vdots$ | $\vdots$ | | $\vdots$ |
| $c_s$ | $a_{s1}$ | $\dots$ | $a_{ss}$ |
| | $b_1$ | $\dots$ | $b_s$ |

**Table A.1:** Butcher Tableau of a general s-stage Runge-Kutta integrator

Using the table above, a general s-stage Runge-Kutta integrator can be described as:

$$y_1 = y_0 + h\sum_{i=1}^{s} b_i k_i. \tag{A.3}$$

where:

$$k_i = f(t_0 + c_i h, y_0 + h\sum_{j=1}^{s} a_{ij} k_j), \quad i = 1, ..., s \tag{A.4}$$

$h$ is the discretization step and where $s$ determines the stage of the integrator. It can be shown that larger integration stages lead to more accurate approximations. Note that if the $a_{ij}$ coefficients consist entirely of non-zero terms then the method is considered implicit. However, if the method is only lower triangular then the method is considered explicit. Implicit methods are integrators that include the function being approximated on both sides of the equation meaning that their solution can only be identified using a root-finding method such as Newton's method. Implicit methods therefore require additional computational resources and can therefore be expensive to use.

## A.3 Symplectic Runge-Kutta Integrators

In Chapter 2, section 2.1.2, we introduced the idea of a symplectic gradient - a gradient along which the Hamiltonian is preserved. Indeed an integrator such as Runge-Kutta can be used to transition a physical system in the direction of the symplectic gradient but such integrators are unable to preserve the symplecticity. In other words, as errors begin to accumulate, non-symplectic integrators compute position and momentum values for which energy drifts away from the original value. However, it can be shown that if a Runge-Kutta method can preserve quadratic first integrals then it is symplectic. This translates to having coefficients from the Butcher-Tableau (Butcher 2008) such that:

$$b_i a_{ij} + b_j a_{ji} = b_i b_j \forall i, j = 1, ...s.$$

Unfortunately, such a constraint enforces the integration scheme to be implicit (Marsden et al. 2001) - a scheme we identified in the previous section as problematic since it requires a root finding approach. To tackle this problem, we investigate variational integrators through the lens of partitioned Runge-Kutta methods.

## A.4 Variational Integrators

To achieve high-order *explicit* symplectic integration, we have to use Partitioned Runge-Kutta (PRK) methods. PRK methods define a separate Butcher Tableau for each variable under investigation, thus, one for position and another for momenta. Using variational calculus on Lagrangians, it can be shown that variational integrators can be described as PRK methods.

From our discussion of earlier work (Chapter 2, section 2.1.3) it is clear that Lagrangian mechanics offers an alternative to the Hamiltonian in generalizing a dynamical system. Rather than position and momentum (canonical coordinates) defining the state space, Lagrangian mechanics is defined using a generalized coordinate state space $(\mathbf{q}, \dot{\mathbf{q}})$. This is particularly useful in physical settings where the description and measurement of generalized coordinates may be easier to work

with than canonical coordinates (Marsden et al. 2001). Given these coordinates, Joseph-Louis Lagrange showed that a scalar value $\mathcal{A}$, referred to as the action, can be defined as the integral of a Lagrangian, $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$:

$$\mathcal{A} = \int_t^{t+1} \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) \mathrm{d}t \tag{A.5}$$

The integral can be thought as inducing multiple paths between points in state space i.e. multiple walks in the domain of $(\mathbf{q}, \dot{\mathbf{q}})$. However, only one path is a stationary state of the action integral. This state lets us move from $t \rightarrow t+1$ with minimal energy. It can be shown, through variational calculus, that this stationary state must satisfy the Euler-Lagrange equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) = \frac{\partial \mathcal{L}}{\partial \mathbf{q}} \tag{A.6}$$

Although complex in form, the action integral and the Euler-Lagrange equations can be discretized and collectively form the basis for variational integrators. Variational Integrators discretize the action integral $\mathcal{A}$ of the Lagrangian $\mathcal{L}$:

$$\mathcal{L}^d(\mathbf{q}_t, \mathbf{q}_{t+1}, h) \approx \int_t^{t+h} \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) \mathrm{d}\tau \tag{A.7}$$

Once discretized, the Euler-Lagrange equations, coupled with the separable Newtonian Lagrangian (Eqn. A.8), can be used to obtain the Störmer-Verlet equation (Eqn. A.9):

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{T}(\dot{\mathbf{q}}) - \mathcal{V}(\mathbf{q}) = \frac{1}{2}\dot{\mathbf{q}}^T M \dot{\mathbf{q}} - \mathcal{V}(\mathbf{q}) \tag{A.8}$$

$$\mathbf{q}_{t+1} = 2\mathbf{q}_t - \mathbf{q}_{t-1} - h^2 M^{-1} \frac{\partial \mathcal{V}(\mathbf{q})}{\partial \mathbf{q}} \tag{A.9}$$

Notice, the Störmer-Verlet equation looks like a discretized second-order differential equation and has a truncation error of $O(h^2)$. The key distinction between this approach and the Hamiltonian is that it allows us to represent information in terms of generalized coordinates and naturally couples the dynamics of momentum with position.

To obtain higher order variational integrators, the work in (Marsden et al. 2001) considers discretizing the Lagrangian by setting the elements of the Lagrangian to polynomials of degree $s$:

$$\mathcal{L}^d(\mathbf{q}_0, \mathbf{q}_1) = h \sum_{i=1}^{s} b_i \mathcal{L}(\mathbf{Q}_i, \dot{\mathbf{Q}}_i) \tag{A.10}$$

where:

$$\mathbf{Q}_i = \mathbf{q}_0 + h \sum_{j=1}^{s} a_{ij} \dot{\mathbf{Q}}_\mathbf{j}, \quad \mathbf{q}_1 = \mathbf{q}_0 + h \sum_{i=1}^{s} b_i \dot{\mathbf{Q}}_i \tag{A.11}$$

If we extremize this Lagrangian with respect to $\dot{\mathbf{Q}}$, (Marsden et al. 2001) show that we obtain a variational integrator which allows us to update position and momentum through the following equations:

$$\mathbf{Q}_i = \mathbf{q}_0 + h \sum_{j=1}^{s} a_{ij} \dot{\mathbf{Q}}_\mathbf{j}, \quad \mathbf{P}_i = \mathbf{p}_0 + h \sum_{j=1}^{s} \hat{a}_{ij} \dot{\mathbf{P}}_j \tag{A.12}$$

$$\mathbf{q}_1 = \mathbf{q}_0 + h \sum_{i=1}^{s} b_i \dot{\mathbf{Q}}_i, \quad \mathbf{p}_1 = \mathbf{p}_0 + h \sum_{i=1}^{s} b_i \dot{\mathbf{P}}_i \tag{A.13}$$

The result of extremizing the integral in this way results in a set of update rules that can be described by a Partitioned Runge-Kutta (PRK) method. It can also be shown that the equations provide a generalized higher-order form of the Störmer-Verlet equation which allows us to couple momentum and position updates with increased accuracy. These update rules are general and show that variational integrators can be derived directly from Lagrangian mechanics. Furthermore, the integration scheme provided can be summarized by PRK methods which are defined by tables such as A.2. A set of coefficients can be chosen such that they enforce symplecticity (Marsden et al. 2001). Furthermore, if the Hamiltonian under investigation is separable, then the method can be made explicit.

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \dots & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array} \quad , \quad \begin{array}{c|ccc} \hat{c}_1 & \hat{a}_{11} & \dots & \hat{a}_{1s} \\ \vdots & \vdots & & \vdots \\ \hat{c}_s & \hat{a}_{s1} & \dots & \hat{a}_{ss} \\ \hline & \hat{b}_1 & \dots & \hat{b}_s \end{array}$$

**Table A.2:** PRK Butcher Tableau

| | | | | |
|---|---|---|---|---|
| $c_1$ | a1 | 0 | 0 | 0 |
| $c_2$ | a1 | a2 | 0 | 0 |
| $c_3$ | a1 | a2 | a3 | 0 |
| $c_4$ | a1 | a2 | a3 | a4 |
| | a1 | a2 | a3 | a4 |

,

| | | | | |
|---|---|---|---|---|
| $c_1$ | 0 | 0 | 0 | 0 |
| $c_2$ | d1 | 0 | 0 | 0 |
| $c_3$ | d1 | d2 | 0 | 0 |
| $c_4$ | d1 | d2 | d3 | 0 |
| | d1 | d2 | d3 | d4 |

**Table A.3:** Yoshida 4th order

| coefficient | value |
|---|---|
| d1 | 0.515352837431122936 |
| d2 | -0.085782019412973646 |
| d3 | 0.441583023616466524 |
| d4 | 0.128846158365384185 |
| a1 | 0.134496199277431089 |
| a2 | -0.224819803079420806 |
| a3 | 0.756320000515668291 |
| a4 | 0.334003603286321425 |

**Table A.4:** Mcate 4th order coefficients

One such explicit symplectic PRK is the fourth order Yoshida method (Yoshida 1990). Their PRK coefficients can be summarized by Table.A.3 where: $w1 = \frac{1}{2-2^{1/3}}$ and $w0 = \frac{-2^{1/3}}{2-2^{1/3}}$ then, $a1 = a4 = w1/2$, $a2 = a3 = (w0 + w1)/2$ and $d1 = d3 = w1$,$d2 = w0$ and $d4 = 0$. Another example is the fourth order Mcate method (Forest et al. 1990) which can be described by PRK coefficients as in Table A.4.

## A.5 Experiments

In our study of variational integrator graph networks, we sought to embed different integration schemes into neural networks. To do so, we needed to empirically evaluate the performance of these integrators on well-known systems to identify in what regimes symplecticity mattered for such systems. We compare a range of integrators on these well known systems prior and our results are summarized in Figures.A.1 and A.2.

## A. Numerical Integrators

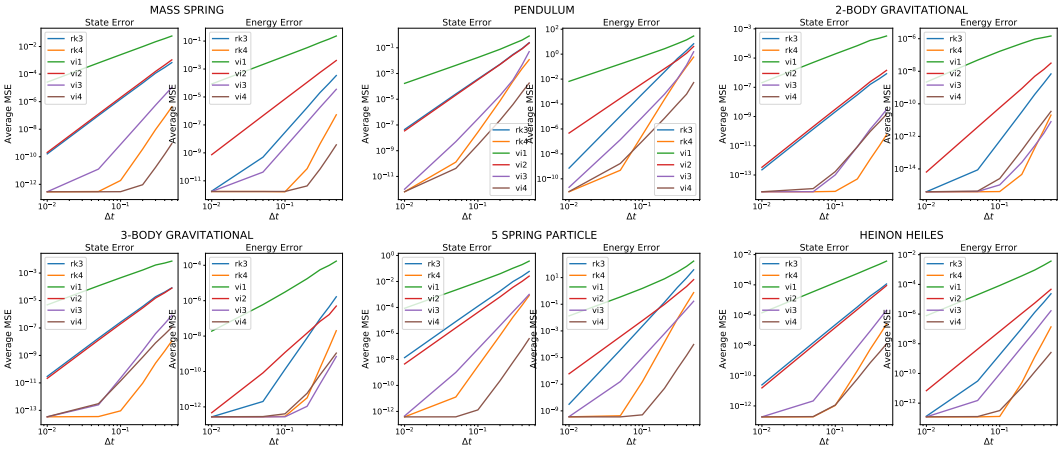



**Figure A.1:** We record the state and energy MSE of over 25 initial conditions for different $\Delta t$ values and average them. Each system is integrated to $T_{max} = 2$. We see that for some systems a fourth order symplectic integrator performs much better than other integrators with larger $\Delta t$. Note, we exclude the RK1 and RK2 plots as they perform very poorly and distract from the performance differences of the higher order methods.

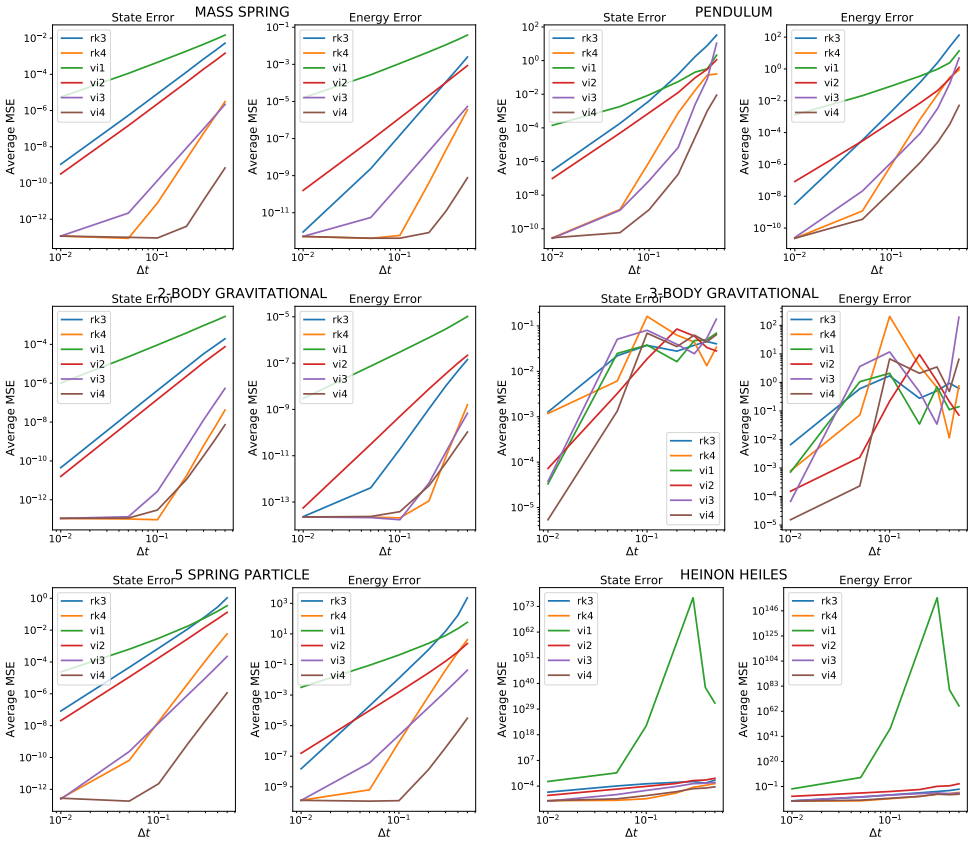



**Figure A.2:** We record the state and energy MSE of over 25 initial conditions for different $\Delta t$ values and average them. Each system is integrated to $T_{max} = 10$.

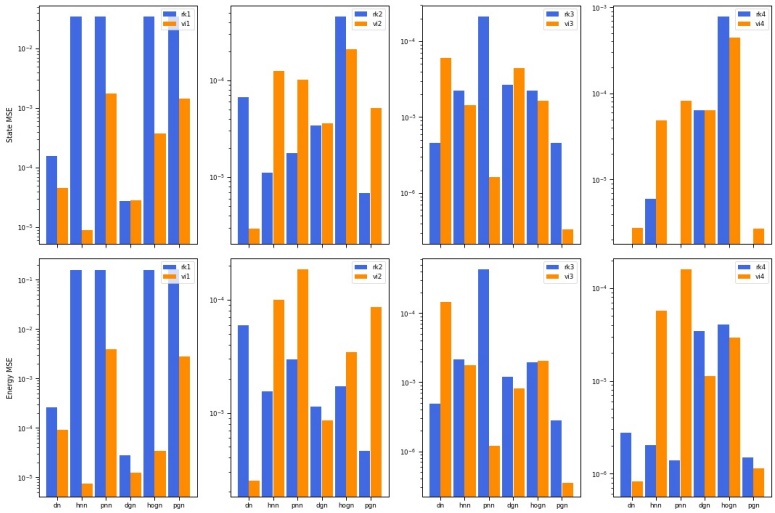
# B
# VIGN Appendices

## B.1 Ablation Results

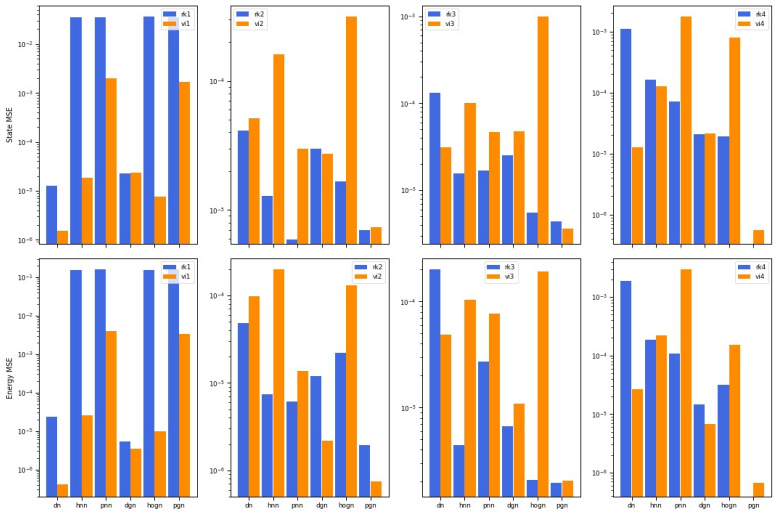
We conduct an extensive ablation across both graph and non-graph based methods with a range of different integration schemes and energy constraints as is described in Ch.3.4. We provide bar charts for the all the experiments in Figures B.1 through B.10.

## B.2 Rollout Results

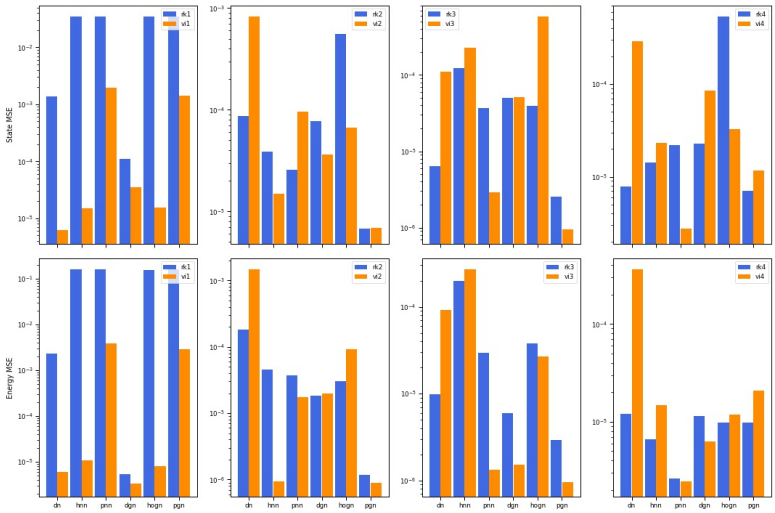
Indeed of all the integrators we embed (up to 4th order), 4th-order integrators lead to the best performance. One way to compare the different networks under a 4th-order integration scheme is to plot the predicted dynamics by these networks for a single test-point. In the figures that follow we highlight the state and energy forecast as well as their MSE values over time compared to ground truth values computed with a RK-8 integration. These results can be found in Figures B.11 through B.20.




**(a)** 2-step integration

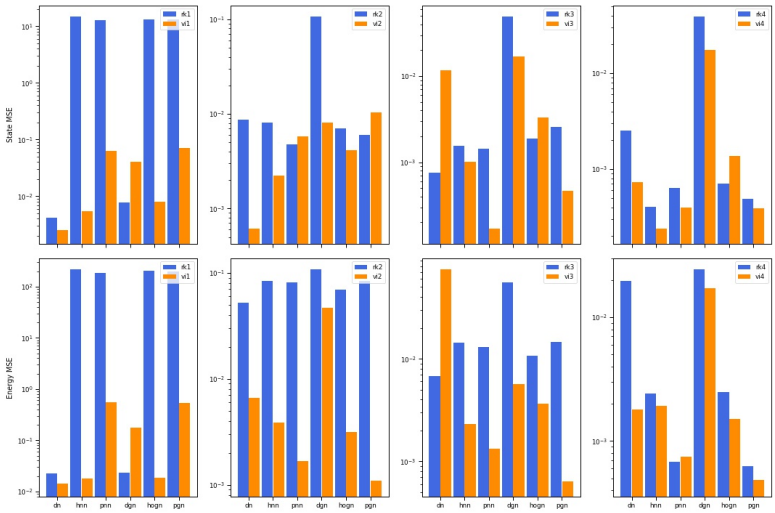



**(b)** 5-step integration

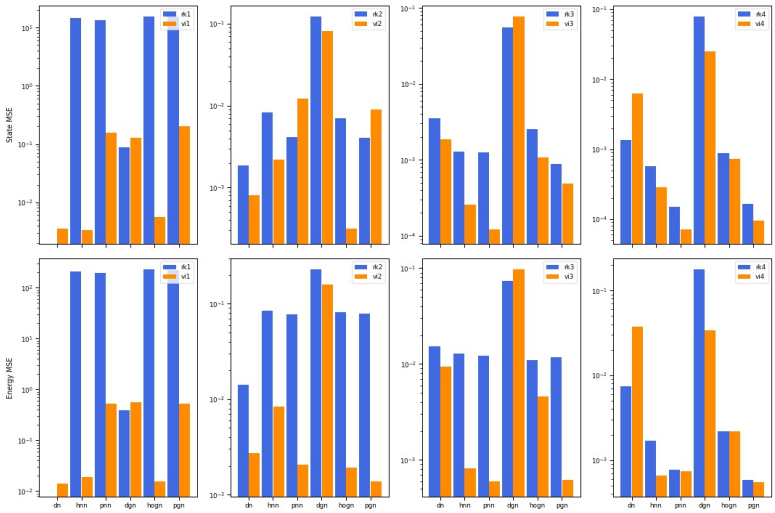



**(c)** 10-step integration

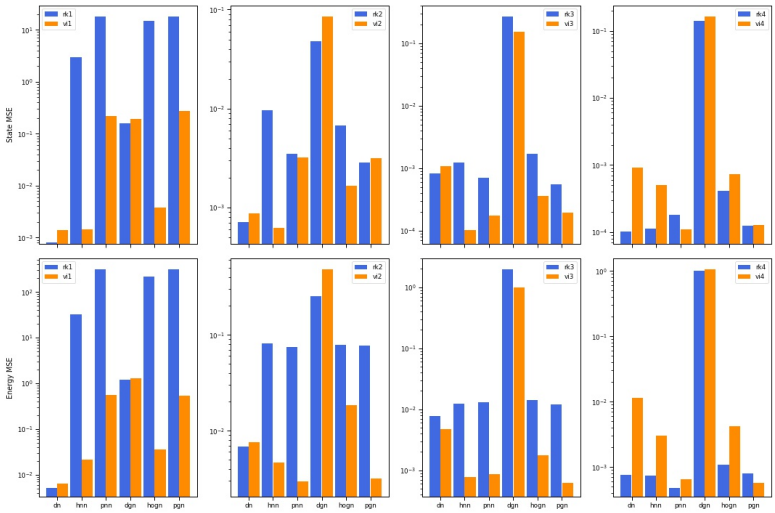
**Figure B.1:** Mass Spring System with noiseless training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

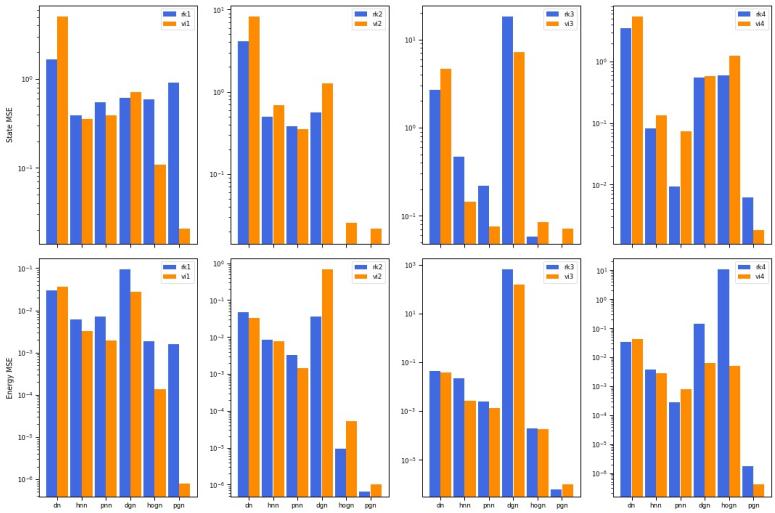



**(b)** 5-step integration

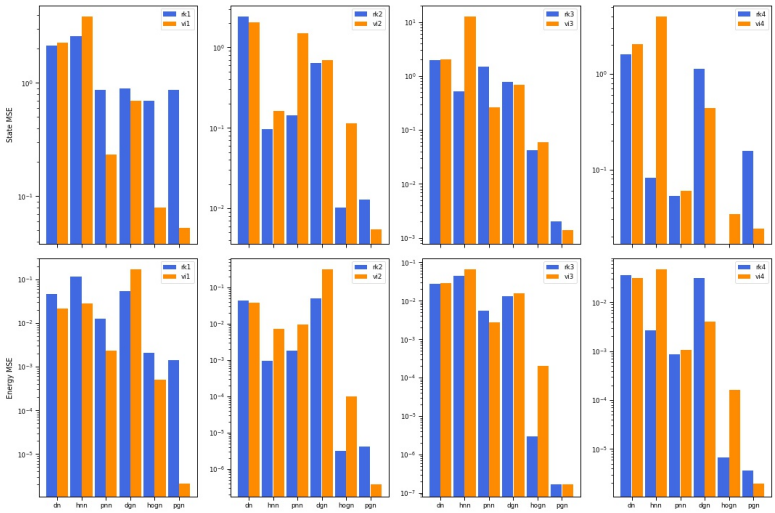



**(c)** 10-step integration

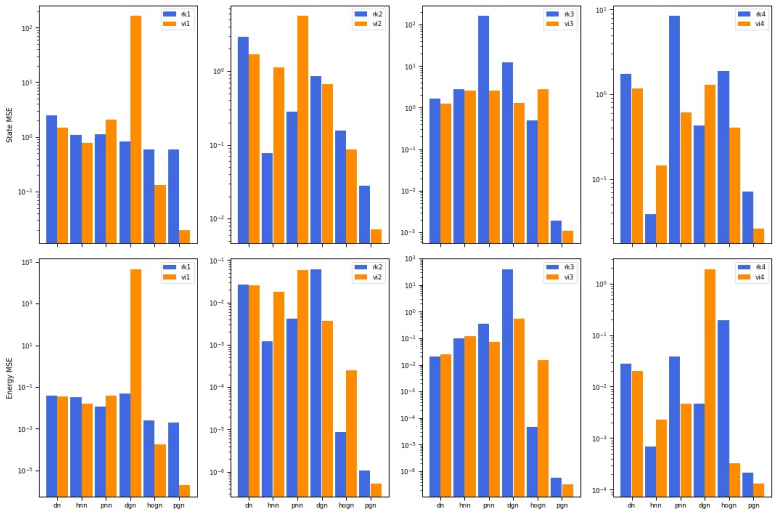
**Figure B.2:** Pendulum with noiseless training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

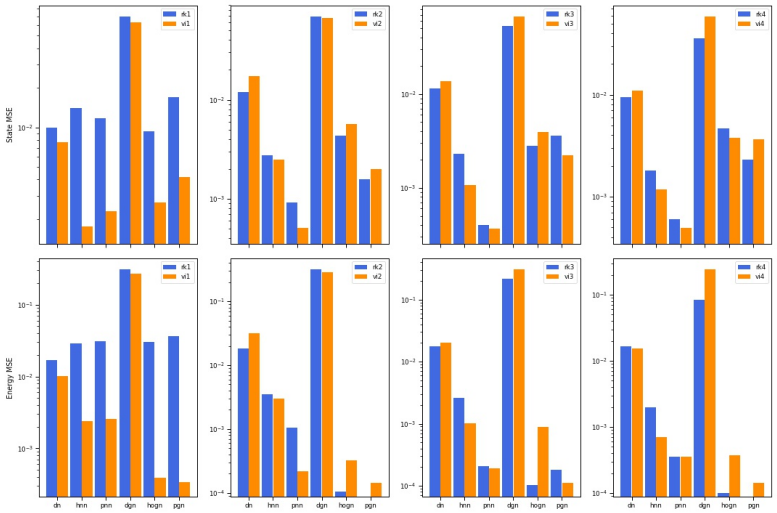



**(b)** 5-step integration

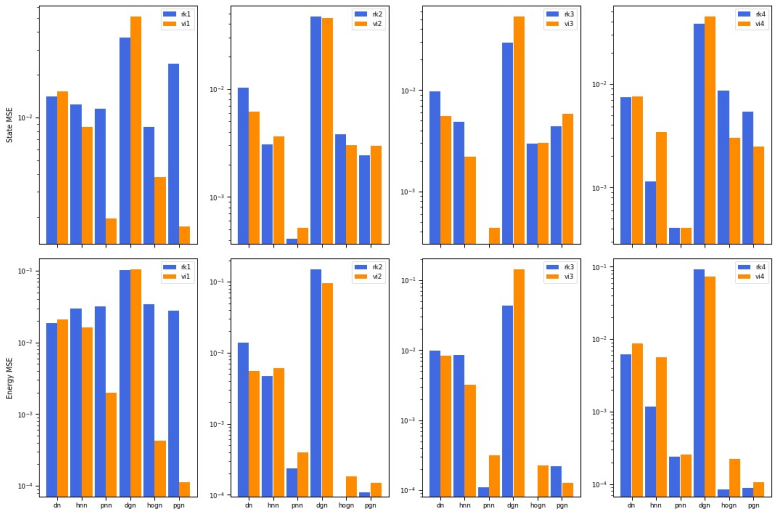



**(c)** 10-step integration

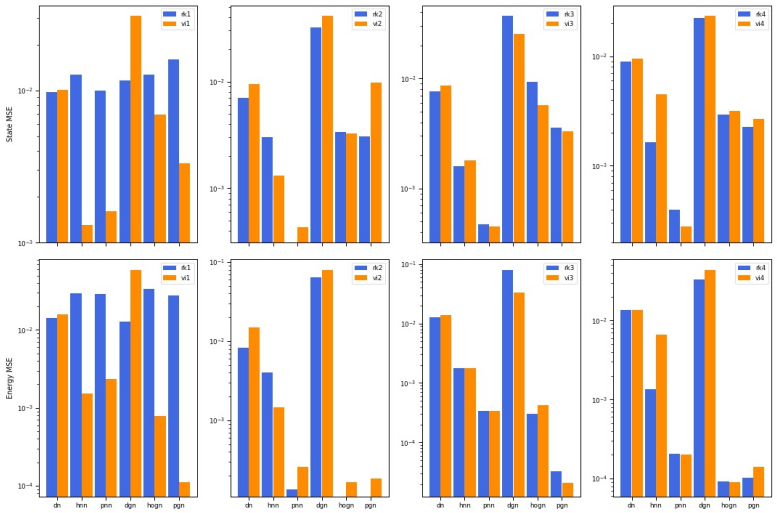
**Figure B.3:** 2-Body gravitational system with noiseless training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

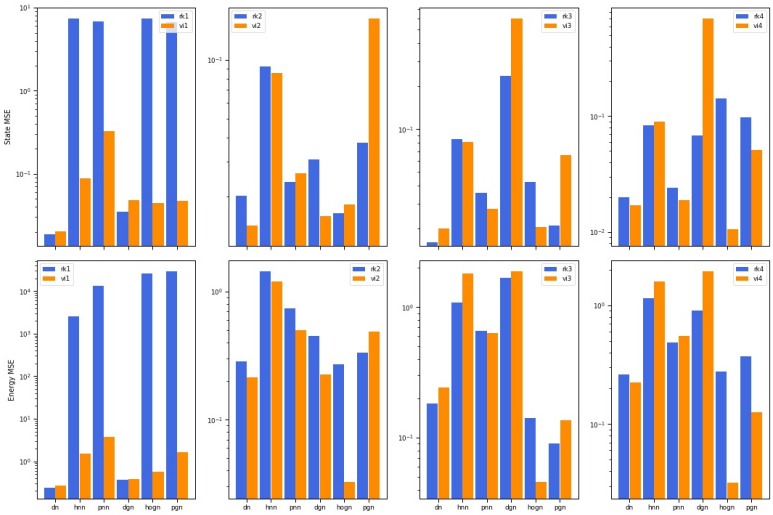



**(b)** 5-step integration

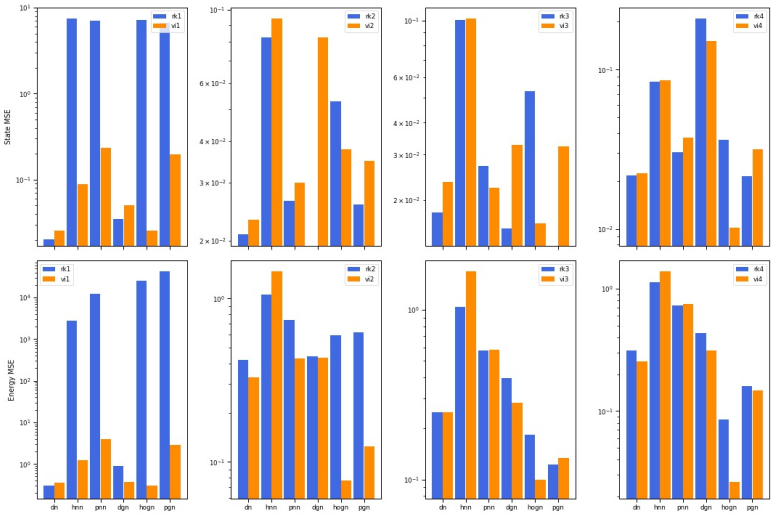



**(c)** 10-step integration

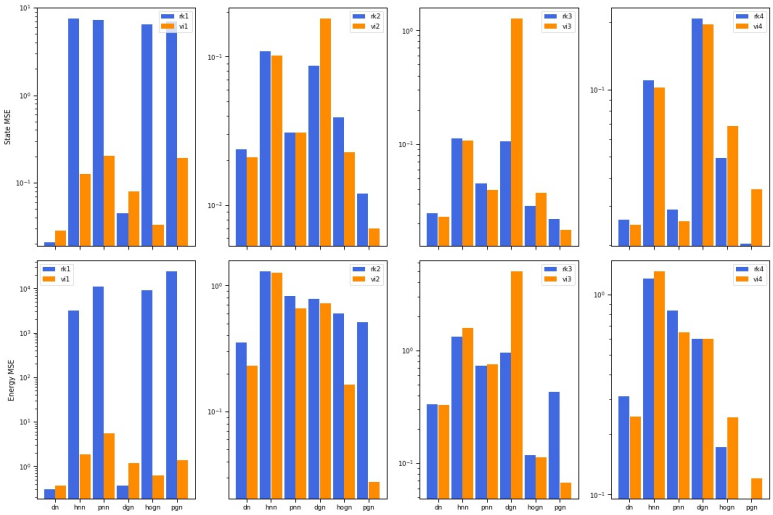
**Figure B.4:** 3-Body gravitational system with noiseless training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

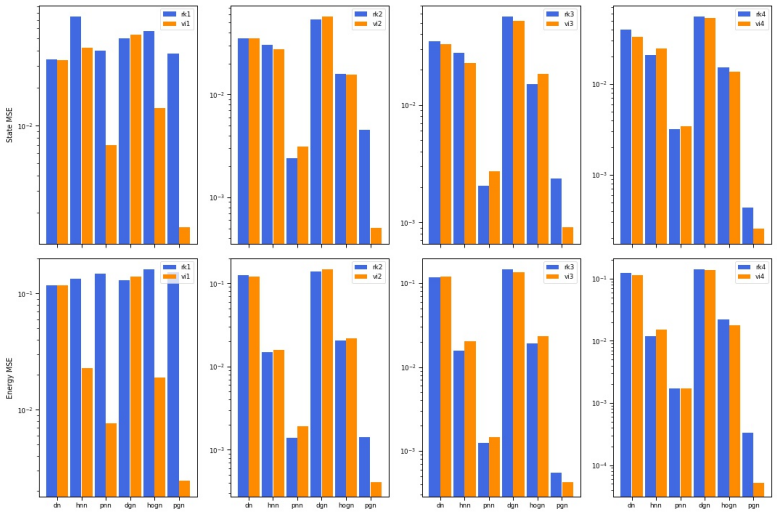



**(b)** 5-step integration

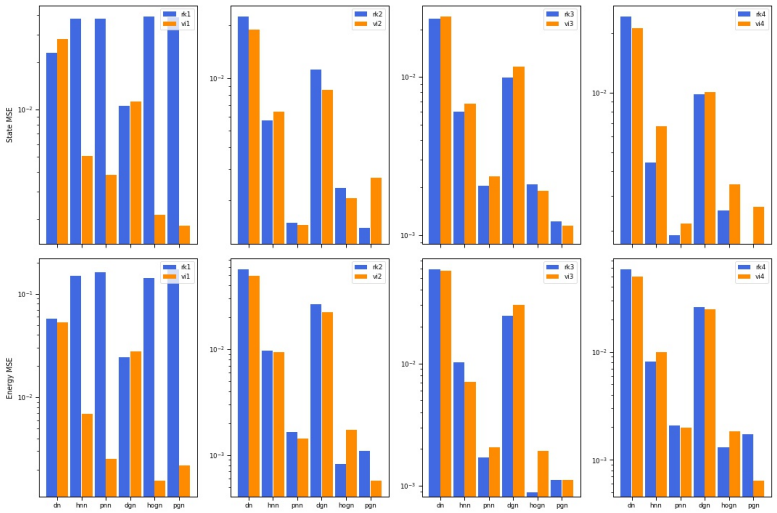



**(c)** 10-step integration

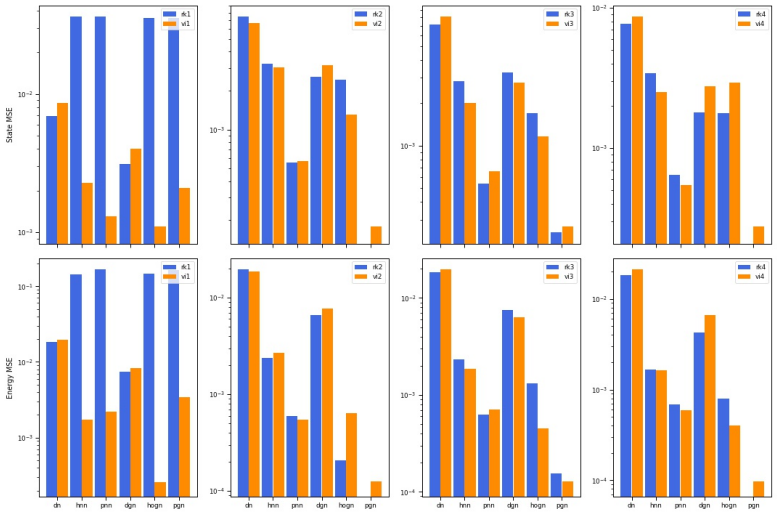
**Figure B.5:** 5-spring particle system with noiseless training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

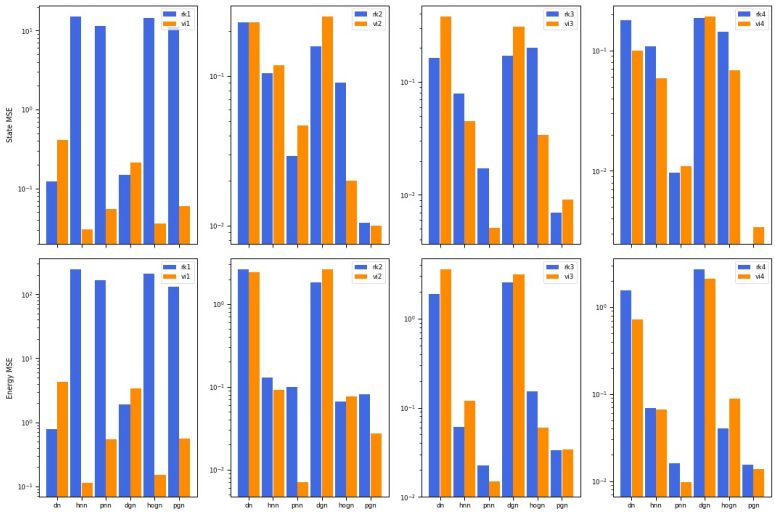



**(b)** 5-step integration

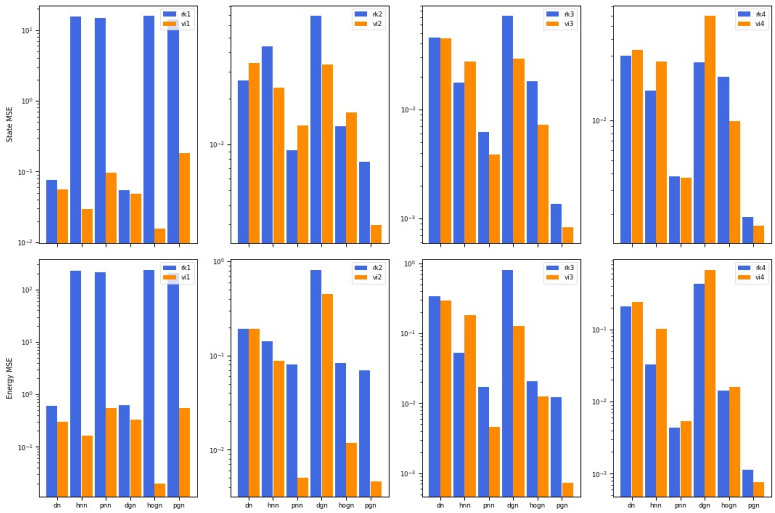



**(c)** 10-step integration

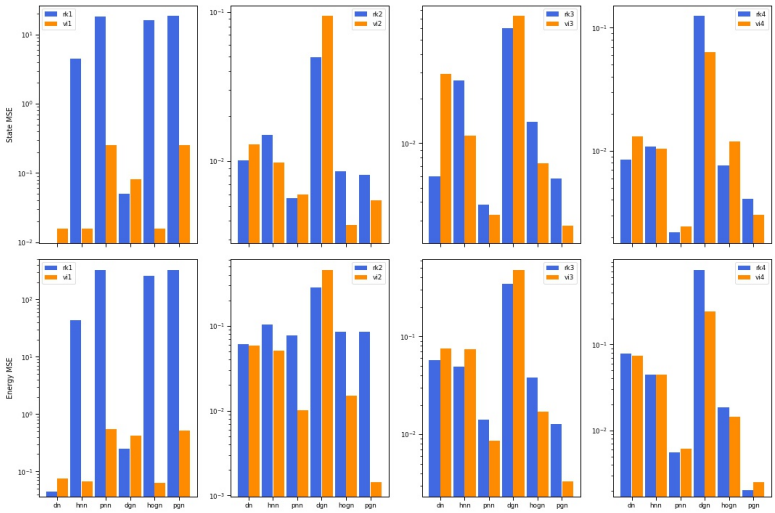
**Figure B.6:** Mass Spring System with noisy training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

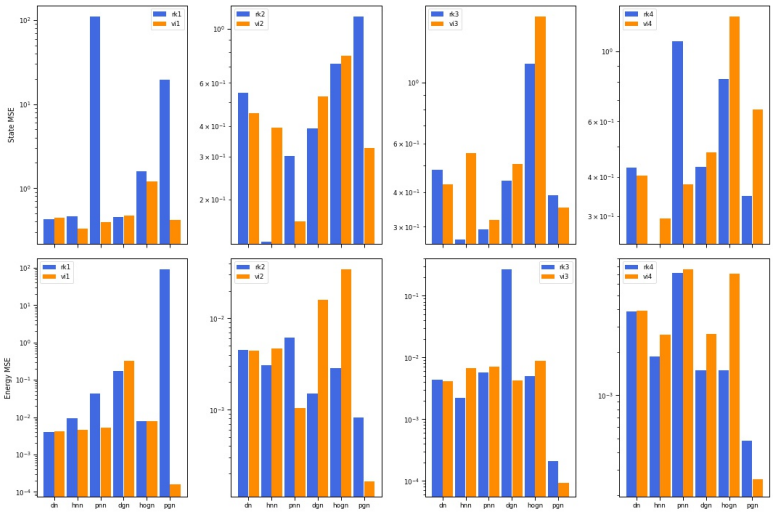



**(b)** 5-step integration

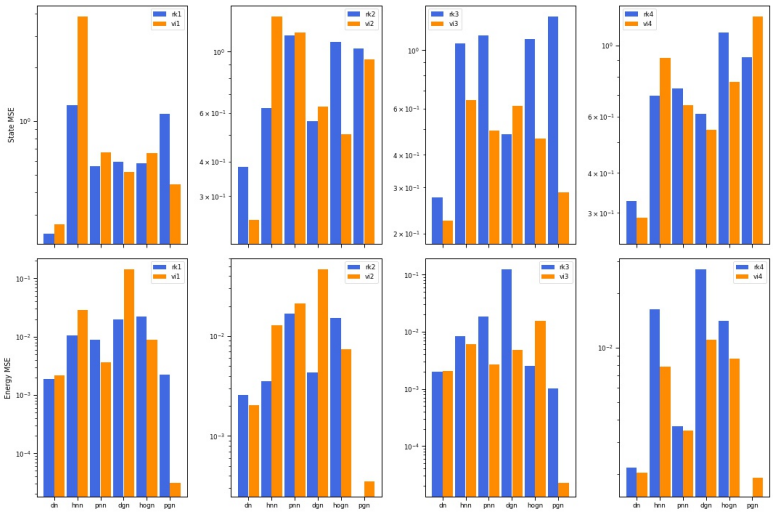



**(c)** 10-step integration

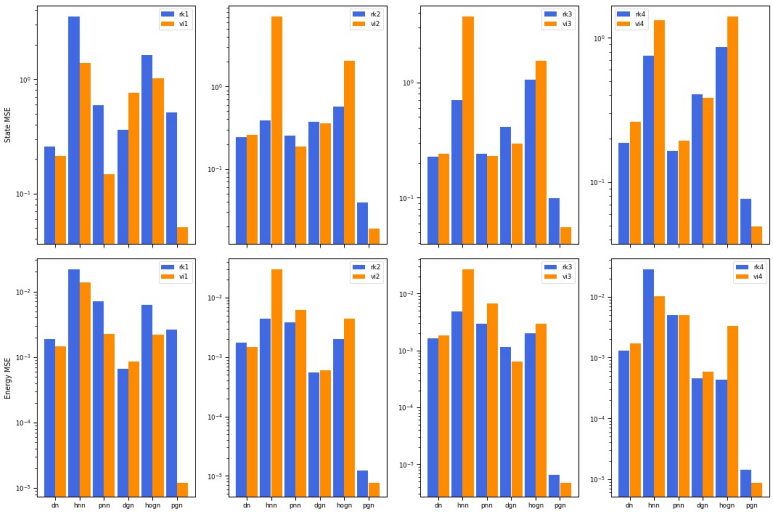
**Figure B.7:** Pendulum with noisy training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

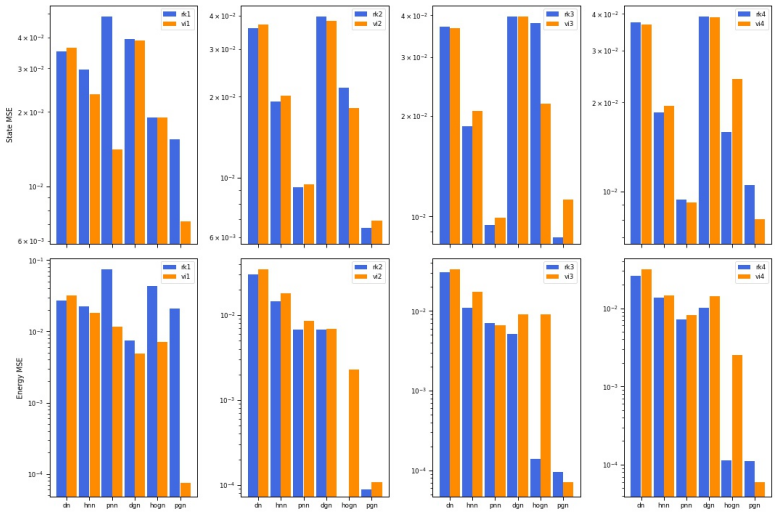



**(b)** 5-step integration

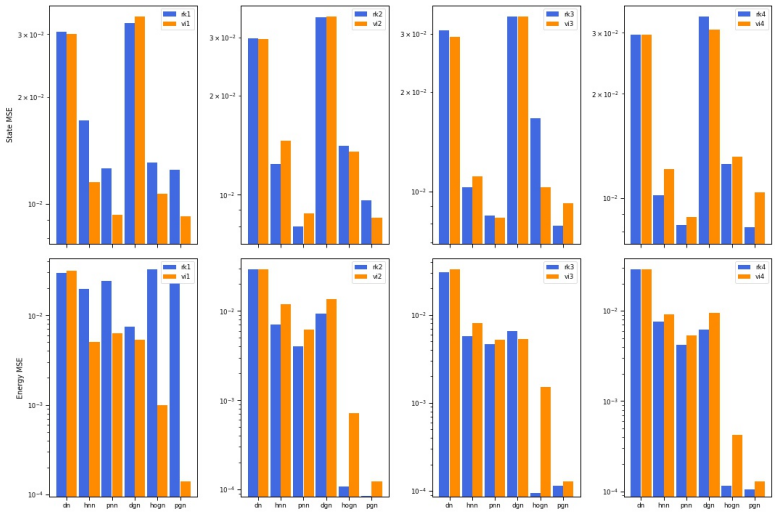



**(c)** 10-step integration

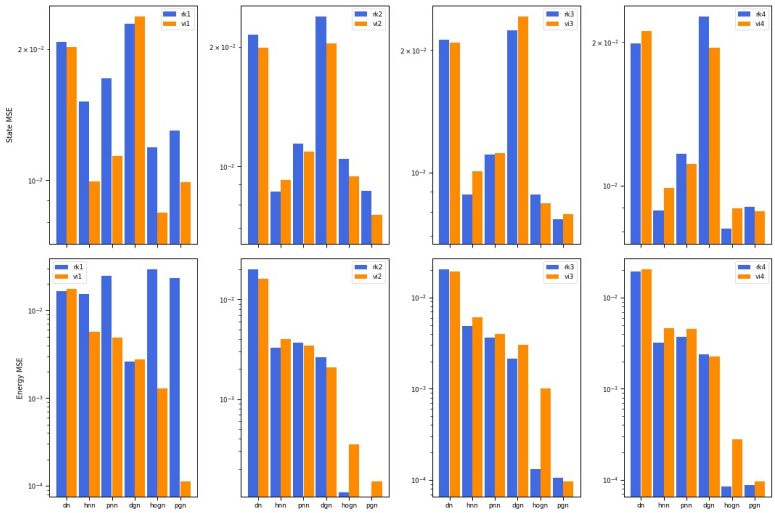
**Figure B.8:** 2-Body gravitational system with noiseless training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

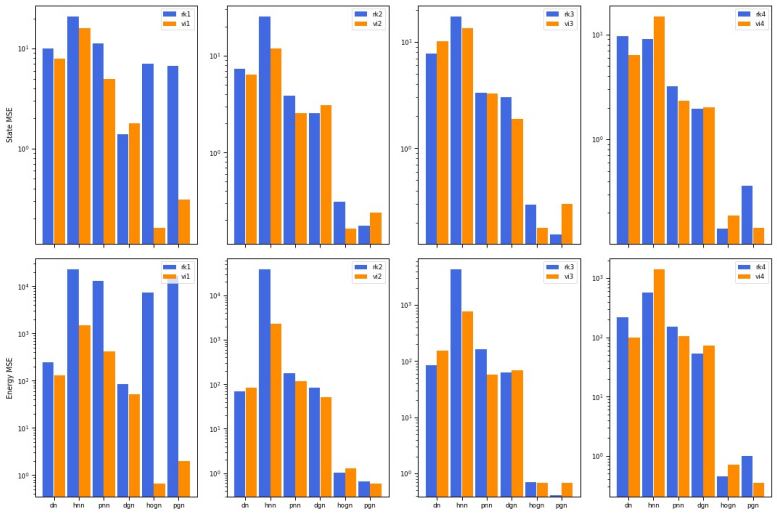



**(b)** 5-step integration

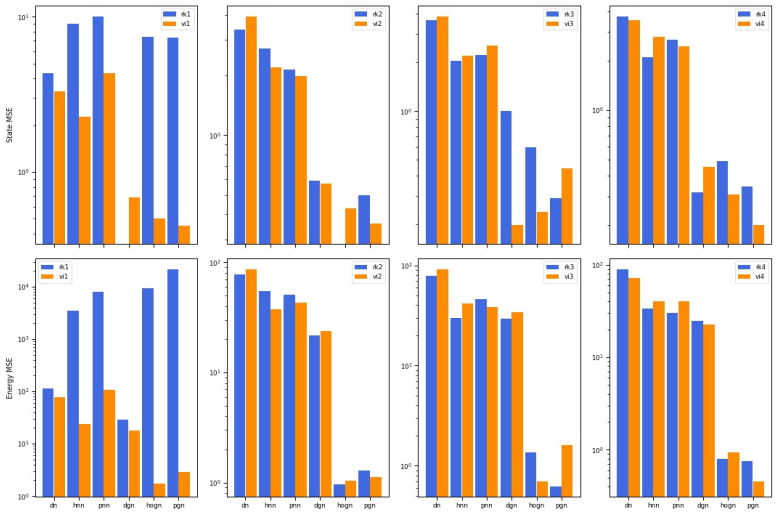



**(c)** 10-step integration

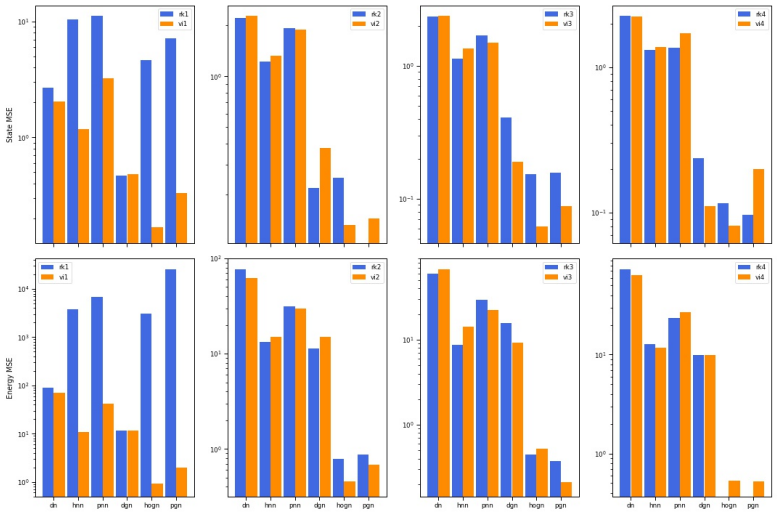
**Figure B.9:** 3-Body gravitational system with noisy training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.




**(a)** 2-step integration

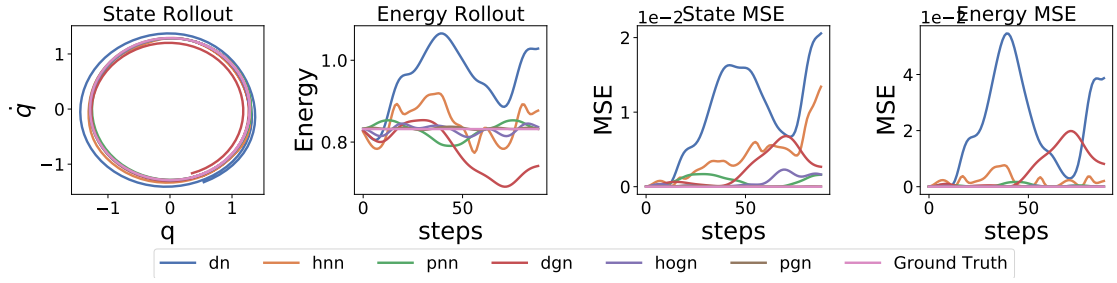



**(b)** 5-step integration

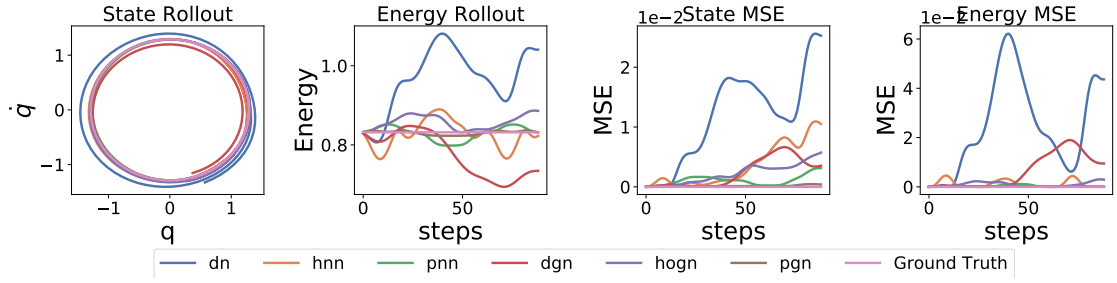



**(c)** 10-step integration

**Figure B.10:** 5-spring particle system with noisy training data. Each bar represents the geometric mean of the MSE of 25 test initial conditions.

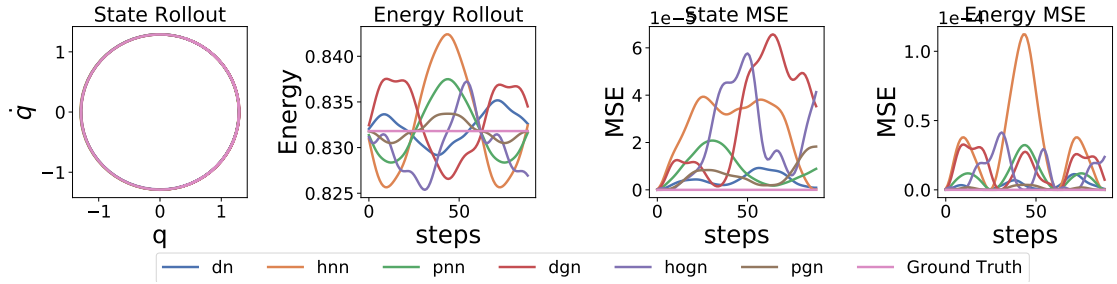



**(a)** RK4 rollout

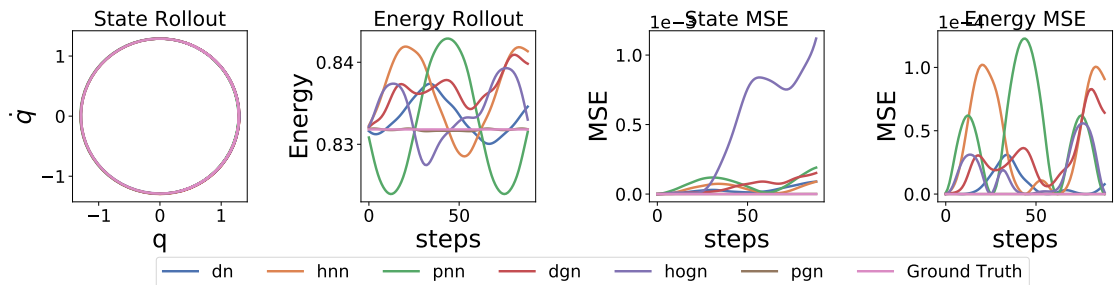



**(b)** VI4 rollout

**Figure B.11:** Rollout of mass-spring system of a single point in the test set. The methods are pretrained with noisy data.

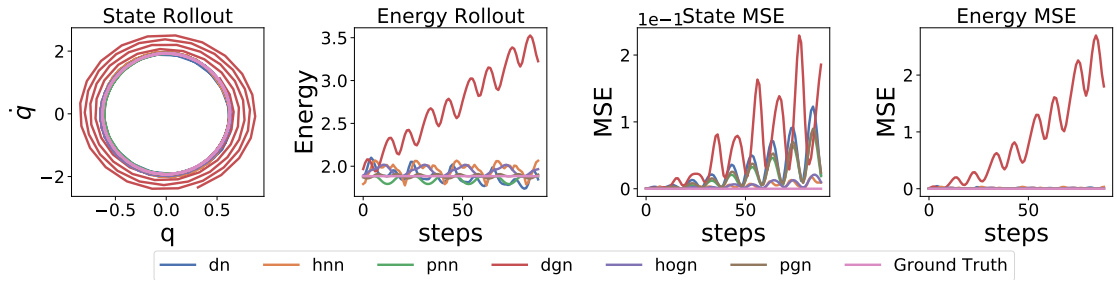



**(a)** RK4 rollout

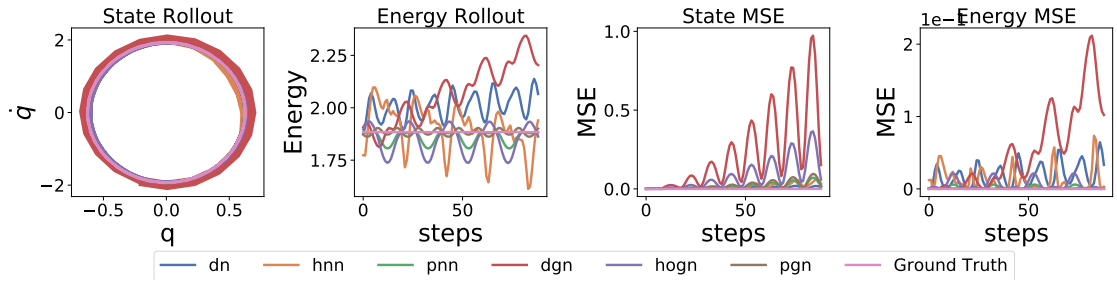



**(b)** VI4 rollout

**Figure B.12:** Rollout of mass-spring system of a single point in the test set. The methods are pretrained with noiseless data.

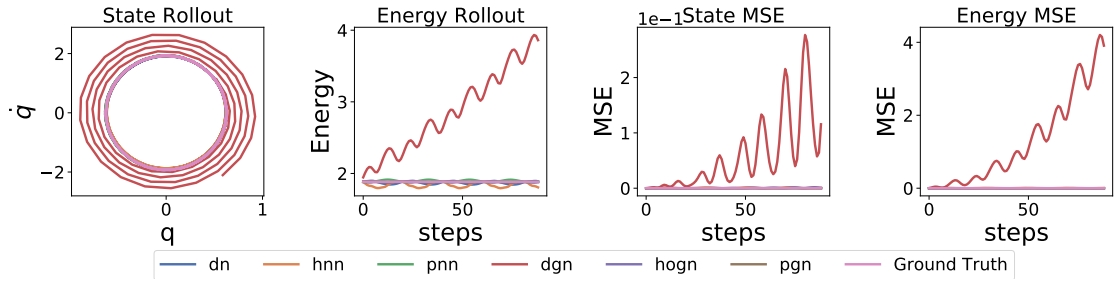



**(a)** RK4 rollout

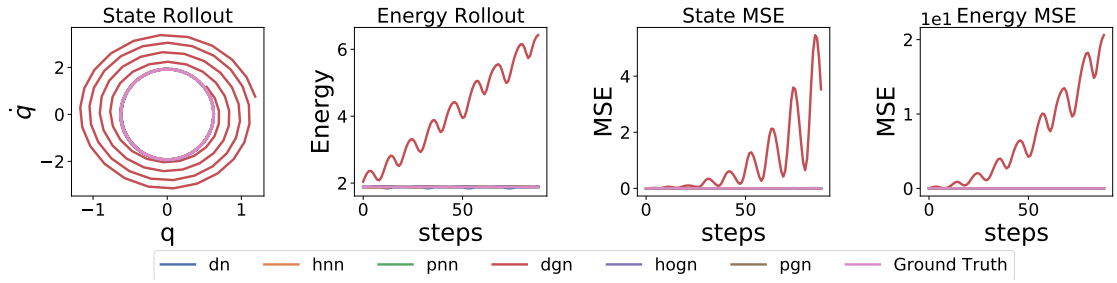



**(b)** VI4 rollout

**Figure B.13:** Rollout of pendulum system of a single point in the test set. The methods are pretrained with noisy data.

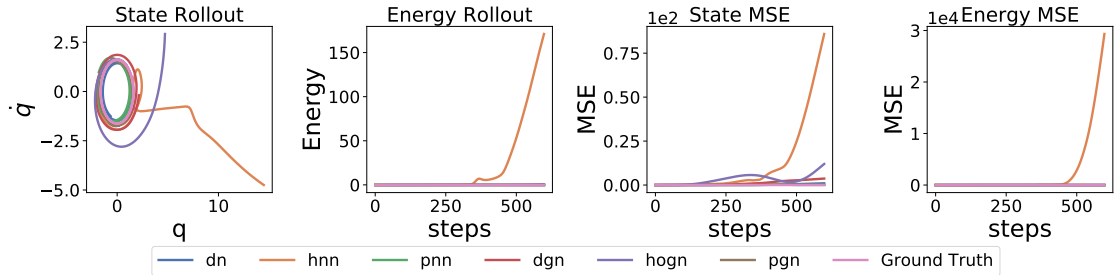



**(a)** RK4 rollout

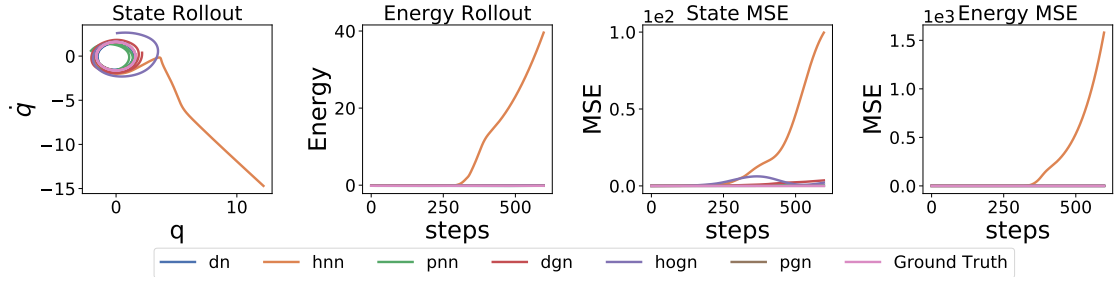



**(b)** VI4 rollout

**Figure B.14:** Rollout of pendulum system of a single point in the test set. The methods are pretrained with noiseless data.

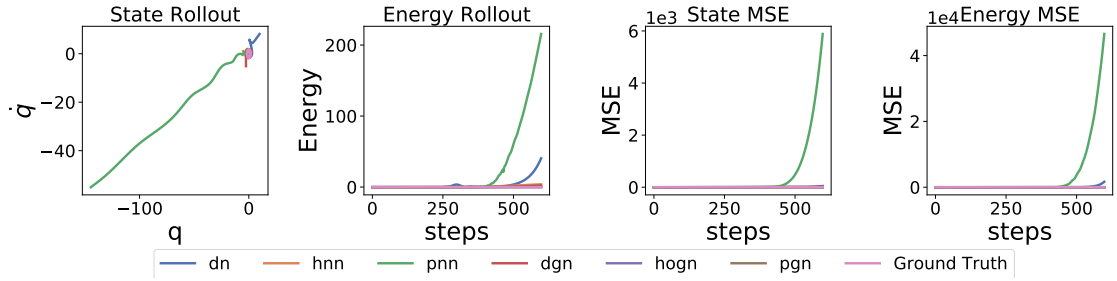



**(a)** RK4 rollout

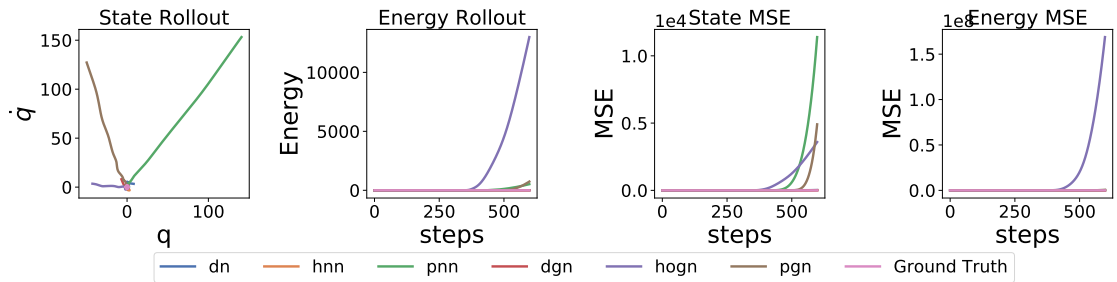



**(b)** VI4 rollout

**Figure B.15:** Rollout of 2-Body gravitational system of a single point in the test set. The methods are pretrained with noisy data.

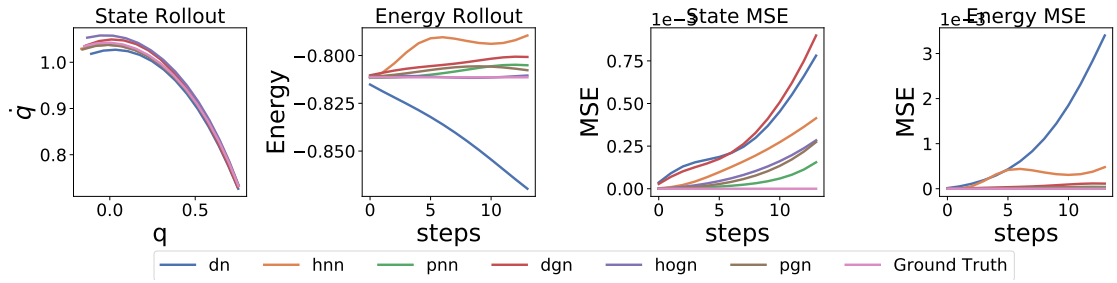



**(a)** RK4 rollout

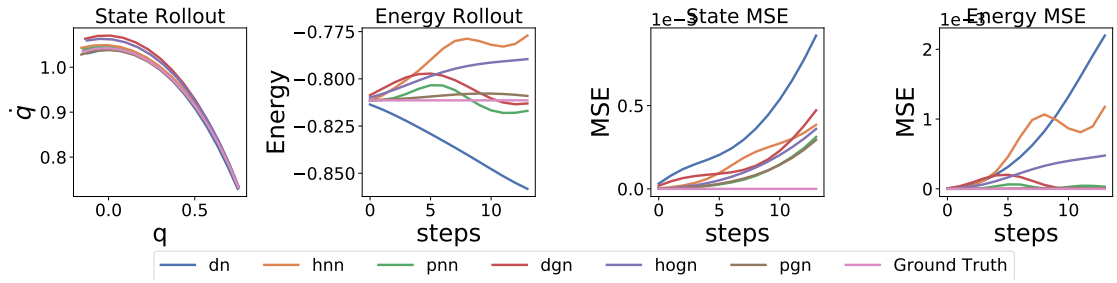



**(b)** VI4 rollout

**Figure B.16:** Rollout of 2-Body gravitational system of a single point in the test set. The methods are pretrained with noiseless data.

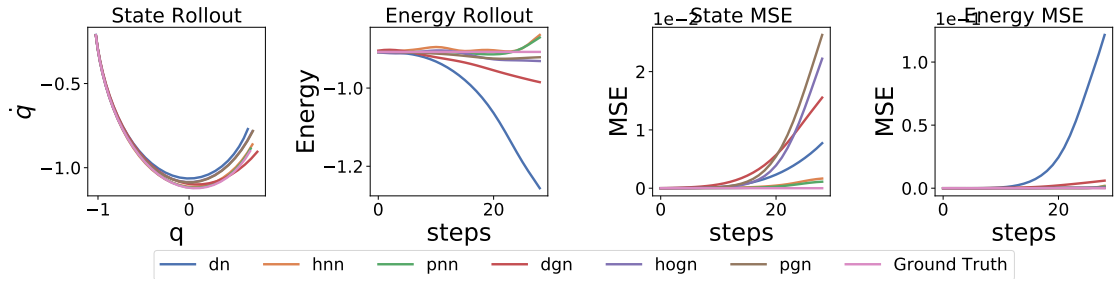



**(a)** RK4 rollout

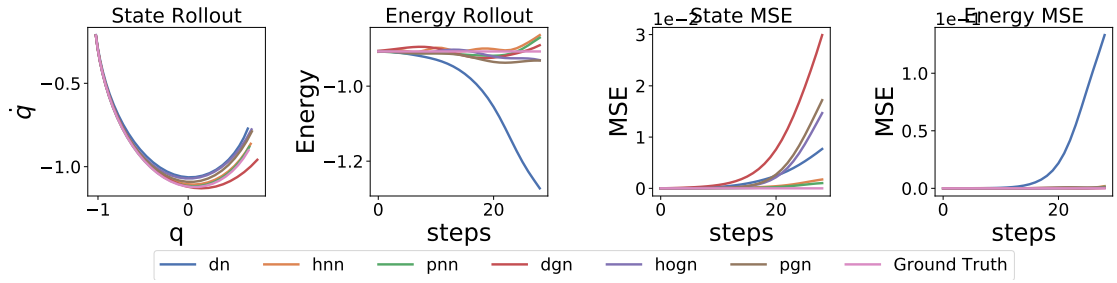



**(b)** VI4 rollout

**Figure B.17:** Rollout of three body gravitational system of a single point in the test set. The methods are pretrained with noisy data.

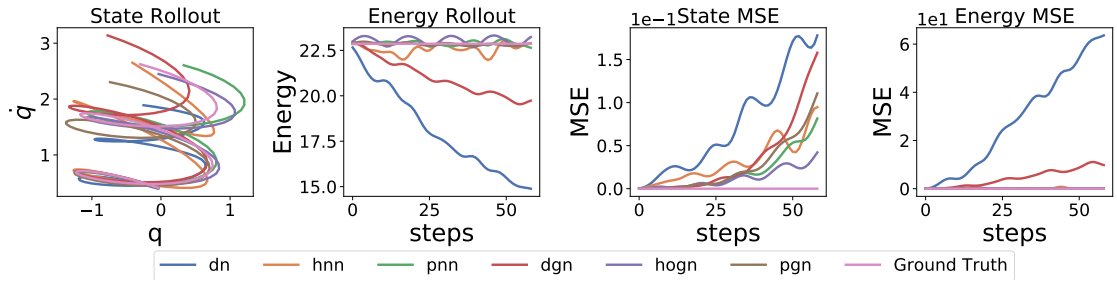



**(a)** RK4 rollout

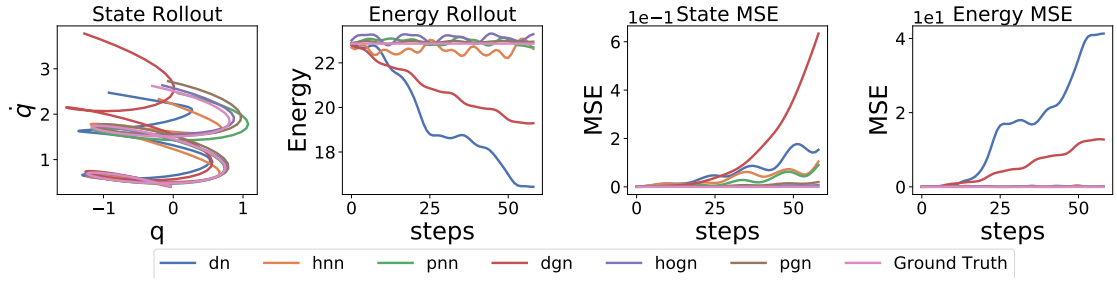



**(b)** VI4 rollout

**Figure B.18:** Rollout of three body gravitational system of a single point in the test set. The methods are pretrained with noiseless data.

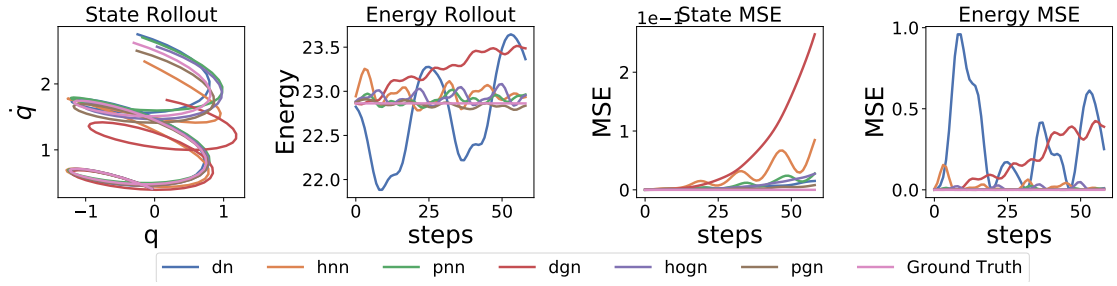



**(a)** RK4 rollout

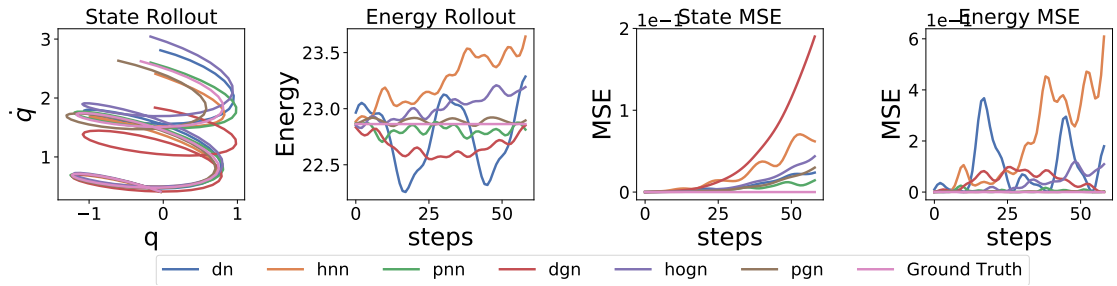



**(b)** VI4 rollout

**Figure B.19:** Rollout of 5 body particle spring system of a single point in the test set. The methods are pretrained with noisy data.




**(a)** RK4 rollout




**(b)** VI4 rollout

**Figure B.20:** Rollout of 5 body particle spring system of a single point in the test set. The methods are pretrained with noiseless data.

# C

# PortHNN Appendices

## C.1 Hyperparameter Optimization

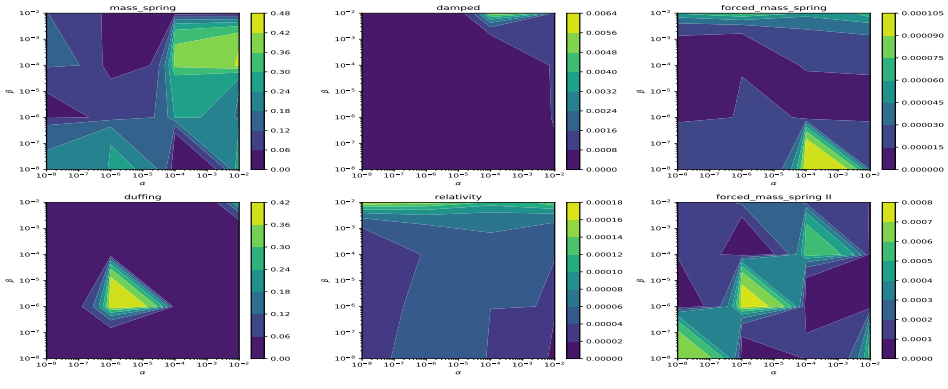



**Figure C.1:** Hyperparameter Optimization for pHNN. We plot, for each system, the validation loss as a function of the $\lambda_F$ and $\lambda_N$ parameters from the loss in eqn. 4.5

The choice of the regularization coefficients $\lambda_F$ and $\lambda_N$ in the loss function 4.5 was made via a grid search across the log-space $[10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}]$, for each hyperparameter, that generates the lowest validation loss (loss plot across parameters plotted in Fig.C.1). Note that since we have limited information during training, this parameter search is necessary as it ensures the Hamiltonian constraint contains most of the information. Without the parameter search, the system is ill-posed and the forcing or damping terms can overfit the data. This result was also verified in (Yin et al. 2021).

## C.2 Pipeline

The detailed procedure of the pipeline for the pHNN model is as follows:

1. Obtain ground truth state variable data $[\mathbf{q}, \mathbf{p}, t]$ and time derivatives $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$ from trajectories of a certain system.

2. Provide state-variable information to pHNN which learns a Hamiltonian, force and damping term to predict $[\hat{\dot{\mathbf{q}}}, \hat{\dot{\mathbf{p}}}]$.

3. Optimize pHNN by minimizing the loss function of Eq. 4.5 in Chapter 4, section 4.3.2.

4. Once trained, use the pHNN in a scientific integrator to evolve a set of random initial conditions in the test set.

5. Assess the performance by calculating the MSE functions of Eqs. 4.6 and 4.7 in Chapter 4, section 4.3.3.

## C.3 Damped Hamiltonians

Here, we illustrate why we cannot include damping in the Hamiltonian. Let us take the following damped system where $\delta$ is the damping coefficient:

$$\ddot{\mathbf{q}} = -\mathbf{q} - \delta\dot{\mathbf{q}}, \tag{C.1}$$

where we know $\mathbf{p} = m\dot{\mathbf{q}}$ which implies $\dot{\mathbf{q}} = m^{-1}\mathbf{p}$.
Then, the integral of the right hand side with respect to q will give us:

$$\frac{\mathbf{q}^T\mathbf{q}}{2} + \delta\mathbf{q}^T\dot{\mathbf{q}}. \tag{C.2}$$

The equation above looks like a modified potential function which can be combined with a kinetic energy term to give a Hamiltonian s.t.:

$$\mathcal{H} = \frac{\mathbf{p}^T\mathbf{p}}{2m} + \frac{\mathbf{q}^T\mathbf{q}}{2} + \delta\mathbf{q}^T\dot{\mathbf{q}}. \tag{C.3}$$

However, although we can recover the differential equation for $\ddot{\mathbf{q}}$ by $-\frac{\partial\mathcal{H}}{\mathrm{d}\mathbf{q}} = -\mathbf{q} - \delta\dot{\mathbf{q}} = \ddot{\mathbf{q}}$, we violate the rule that $\dot{\mathbf{q}} = m^{-1}\mathbf{p}$ since $\frac{\partial\mathcal{H}}{\mathrm{d}\mathbf{p}} = \dot{\mathbf{q}} + \delta\mathbf{q} \neq \dot{\mathbf{q}}$.

# C.4 Tabularized Results

**Table C.1:** State and Energy MSE for each method averaged across 25 initial test points

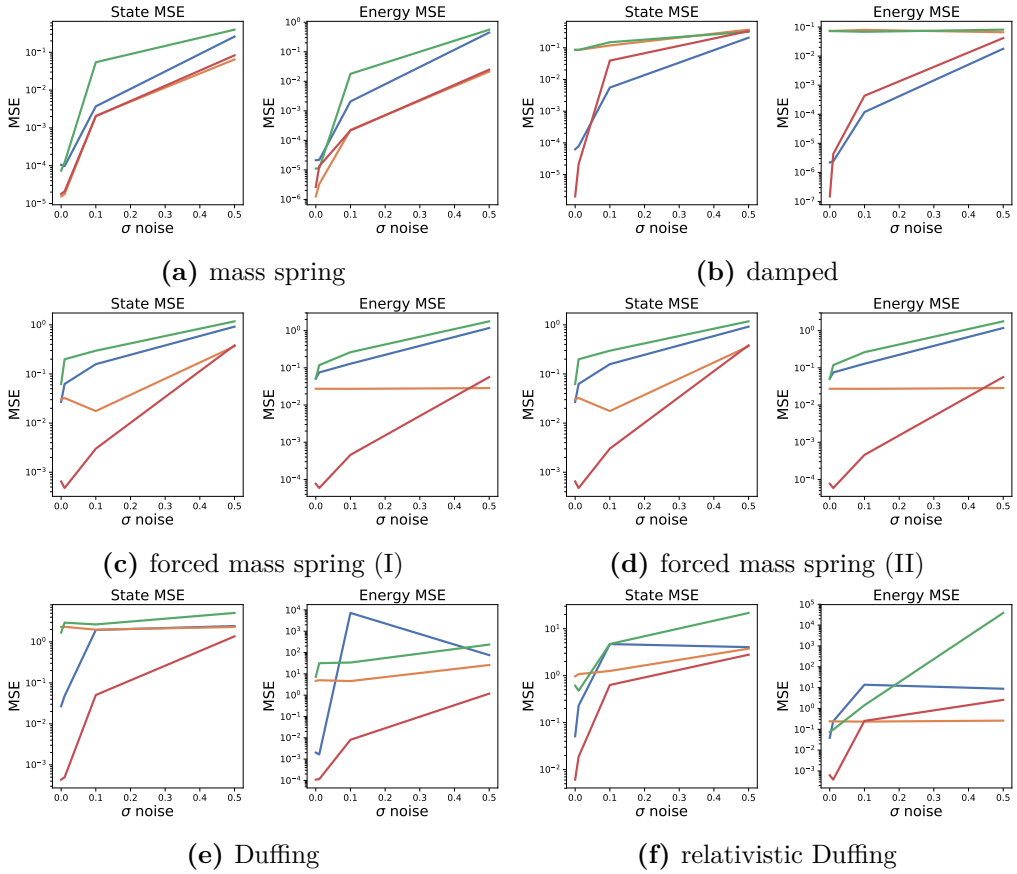
| | Baseline | | HNN | | TDHNN | | pHNN | |
|---|---|---|---|---|---|---|---|---|
| | **State** | **Energy** | **State** | **Energy** | **State** | **Energy** | **State** | **Energy** |
| **Mass Spring** | 1.22E-4 ± 1.06E-4 | 1.56E-5 ± 1.84 E-5 | **1.33E-5 ± 8.05E-6** | **1.30E-6 ± 1.31E-6** | 7.84E-5±1.02E-4 | 9.26E-6 ± 1.13E-5 | 2.91E-5 ± 2.57E-5 | 2.33E-6 ± 2.41E-6 |
| **Damped Mass-Spring** | 6.4E-5 ± 2.904E-5 | 4.68E-7 ± 7.51E-7 | 2.81E-1± 1.37E-1 | 1.81E-1 ± 1.37E-1 | 3.01E-1±1.43E-1 | 1.92E-1±1.42E-1 | **1.74E-6±7.26E-7** | **1.08E-7±1.73E-7** |
| **Forced (I)** | 5.21E-4 ± 5.74E-4 | 7.77E-4 ± 5.52E-4 | 2.36E-1 ± 1.43E-1 | 2.34E-1 ± 1.70E-1 | 2.91E-1 ± 3.06E-1 | 7.61E-1 ± 1.23 | **1.53E-5 ± 1.16E-5** | **9.21E-6 ± 7.11E-6** |
| **Forced (II)** | 3.14E-2 ± 2.54E-2 | 7.00E-2 ± 1.14E-1 | 3.11E-2 ± 2.67E-2 | 2.70E-2 ± 1.12E-2 | 2.22E-1 ± 2.82E-1 | 4.68E-1 ± 1.06 | **4.51E-4 ± 4.85E-4** | **8.91E-5 ± 1.01E-4** |
| **Duffing** | 1.11E-2 ± 2.43E-2 | 8.93E-4 ± 8.75E-4 | 2.31 ± 1.21 | 4.90 ± 5.30 | 2.91 ± 3.01 | 2.43E1 ± 5.11E1 | **3.50E-4 ±3.54E-4** | **5.32E-5 ± 5.38E-5** |
| **Relativistic Duffing** | 5.85E-2 ± 7.05E-2 | 3.61E-2 ± 4.76E-2 | 1.39 ± 1.66 | 2.37E-1 ± 1.55E-1 | 5.9E-1 ± 1.44 | 4.39E-2 ± 5.34E-2 | **4.96E-3 ± 8.61E-3** | **1.28E-3±1.45E-3** |

**Table C.2:** State and Energy MSE for each method embedded with an RK4 integrator averaged across 25 initial test points

| | Baseline | | HNN | | TDHNN | | pHNN | |
|---|---|---|---|---|---|---|---|---|
| | **State** | **Energy** | **State** | **Energy** | **State** | **Energy** | **State** | **Energy** |
| **Mass Spring** | 1.18E-4 ± 1.01E-4 | 1.75E-5 ± 1.90E-5 | **1.96E-5 ± 1.98E-5** | **1.77E-6 ± 1.79E-6** | 2.78E-5 ± 2.95E-5 | 1.56E-5 ± 1.79E-5 | 2.78E-5 ± 2.65E-5 | 2.69E-6 ± 2.76E-6 |
| **Damped Mass-Spring** | **1.95E-4 ± 8.77E-5** | **8.11E-7 ± 7.49E-7** | 2.80E-1 ± 1.49E-1 | 1.81E-1 ± 1.36E-1 | 3.06E-1 ± 1.46E-1 | 1.94E-1 ± 1.43E-1 | 2.70E-3 ± 1.34E-3 | 7.37E-4 ± 5.85E-4 |
| **Forced (I)** | 6.82E-4 ± 8.30E-4 | 1.31E-3 ± 4.25E-3 | 1.84E-1 ± 1.15E-1 | 2.27E-1 ± 1.66E-1 | 7.62E-4 ± 5.39E-4 | 8.23E-4 ± 6.17E-4 | **3.45E-5 ± 2.32E-5** | **8.54E-6 ± 8.27E-6** |
| **Forced (II)** | 2.49E-2 ± 1.29e-2 | 3.52E-2 ± 1.12e-2 | 2.80E-2 ± 2.18E-2 | 2.76E-2 ± 1.13E-2 | 5.01E-2 ± 5.11E-2 | 8.02E-2 ± 5.83E-2 | **1.33E-3 ± 1.64E-3** | **1.63E-4 ± 1.78E-4** |
| **Duffing** | 5.41E-2 ± 1.47E-1 | 2.41E-3 ± 2.55E-3 | 2.29E0 ± 1.19E0 | 4.54E0 ± 4.58E0 | 1.84E0 ± 1.02E0 | 5.11E0 ± 5.58E0 | **3.57E-3 ± 2.99E-3** | **7.10E-4 ± 1.17E-3** |
| **Relativistic Duffing** | 1.70E-2 ± 1.79E-2 | 2.16E-2 ± 3.04E-2 | 1.91E-1 ± 1.59E-1 | 5.83E-2 ± 4.19E-2 | 3.18E-2 ± 2.20E-2 | 4.83E-2 ± 4.62E-2 | **1.18E-3 ± 1.95E-3** | **6.09E-4 ± 1.08E-3** |

# C.5 Results with Noise

During the training of the HNN (Samuel Greydanus et al. 2019), the authors add Gaussian noise with a standard deviation $\sigma = 0.1$ to the input state vector data. The reason this is done is to ensure the model is robustly trained. We run a set of experiments to test the robustness to this 'noisy' input. The results for each system investigated can be found in Fig.C.2.




**(a)** mass spring **(b)** damped

**(c)** forced mass spring (I) **(d)** forced mass spring (II)

**(e)** Duffing **(f)** relativistic Duffing

**Figure C.2:** Results across systems when a Gaussian noise with standard deviation $\sigma$ is added to the position and momentum data

## C.6 2-body coupled spring system

We run an additional study of a 2-body system with masses $m$ coupled by springs with constants $k$. The system is on a horizontal plane and consists of two masses, each attached to fixed walls on the left and right respectively and coupled to each other by another spring. A driving force of $\cos(t)$ is applied to the first mass. The Hamiltonian for this system is:

$$\mathcal{H} = \frac{p_1^2}{2m} + \frac{p_2^2}{2m} + kq_1^2 + kq_2^2 - kq_1q_2 - q_1\cos(t)$$

The initial conditions are sampled such that $\mathbf{q} \in [-0.5, 0.5]^2$ and $\mathbf{p} \in [-0.2, 0.2]^2$. $\Delta t = 0.1$ and $T_{\text{max}} = 5$. Only 25 initial conditions are used to train the system. The results are summarized in Fig.C.3.

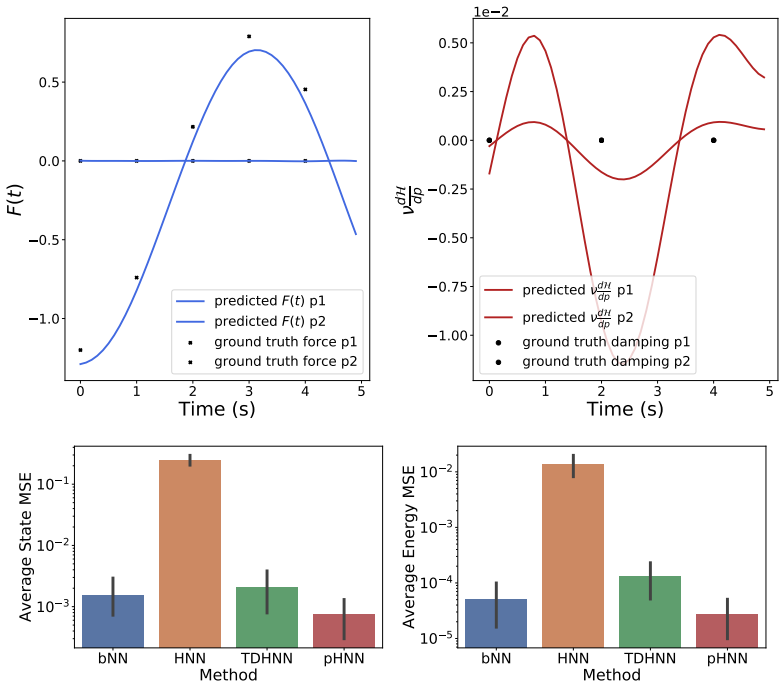



**Figure C.3:** spring coupled results: pHNN performs the best with bNN performing similarly well. Results are reported for an embedded integrator.

# D

# Transfer Differential Equation Appendices

## D.1 System of second-order differential equations

To compute the analytic $W_{\text{out}}$ for a system of second-order differential equations we begin by defining:

$$\ddot{\boldsymbol{\psi}} = A\boldsymbol{\psi}$$

where dots denote time derivatives and

$$\boldsymbol{\psi} = \begin{bmatrix} H & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} W_q \\ W_p \end{bmatrix}.$$

Then, by computing the $L2$ loss of $\ddot{\boldsymbol{\psi}} - A\boldsymbol{\psi}$ and the initial conditions $\psi_0 = \psi(0), \dot{\psi}_0 = \dot{\psi}(0)$ and differentiating it with respect to $W_{\text{out}} = [W_q, W_p]$ we obtain:

$$W_{\text{out}} = \left(D_H^T D_H + H_0^T H_0 + H_{0d}^T H_{0d}\right)^{-1} \left(H_0^T \psi_0 + H_{0d}^T \dot{\psi}_0\right)$$

where

$$D_H = \begin{bmatrix} \ddot{H} & 0 \\ 0 & \ddot{H} \end{bmatrix},$$
$$H_0 = \begin{bmatrix} H(0) & 0 \\ 0 & H(0) \end{bmatrix},$$
$$H_{0d} = \begin{bmatrix} \dot{H}(0) & 0 \\ 0 & \dot{H}(0) \end{bmatrix}.$$

## D.2 QR Decomposition

In Chapter 5, section 5.4.1, we define the loss function for ODEs as:

$$\mathcal{L} = (\hat{D}_n u_\theta(t) - f(t))^2 + (\bar{D}_0 u_\theta(t) - u_{\text{ic}})^2, \tag{D.1}$$

For ease of notation, let $\hat{D}_n u_\theta(t) = \hat{Y}$, $f(t) = Y$, $\bar{D}_0 u_\theta(t) = \hat{Y}_0$ and $u_{\text{ic}} = Y_0$. The loss function above can be re-written as a single loss function s.t.:

$$\mathcal{L} = \left( \begin{bmatrix} \hat{Y} \\ \hat{Y}_0 \end{bmatrix} - \begin{bmatrix} Y \\ Y_0 \end{bmatrix} \right)^2. \tag{D.2}$$

To see this, we can expand the vector notation as:

$$\mathcal{L} = \left( \begin{bmatrix} \hat{Y} & \hat{Y}_0 \end{bmatrix} - \begin{bmatrix} Y & Y_0 \end{bmatrix} \right) \left( \begin{bmatrix} \hat{Y} \\ \hat{Y}_0 \end{bmatrix} - \begin{bmatrix} Y \\ Y_0 \end{bmatrix} \right), \tag{D.3}$$

which when expanded resolves to Eqn.D.1.

By differentiating the above loss equation with respect to $W_{\text{out}}$ and setting it to zero, we obtain a linear least squares problem as

$$\left( H^T H + H_0^T H_0 \right) W_{\text{out}} = \left( H^T Y + H^T Y_0 \right), \tag{D.4}$$

which is of the form

$$A^T A \; W_{\text{out}} = A^T Y. \tag{D.5}$$

Since the left hand-side is a square matrix $A^T A$, it is possible to take its pseudo-inverse. However, for a number of problems, it is possible that the matrix $A$ has a large condition number and therefore the squaring procedure of the normal equations squares the condition number making it unstable. Although it is possible to take a pseudo-inverse, in such cases, rather than using the normal equations to obtain a solution to $W_{\text{out}}$, it is possible to solve for $W_{\text{out}}$ using QR decomposition. In other words:

$$A W_{\text{out}} = Y, \tag{D.6}$$

and since $A$ is not square, we can decompose it into $A = QR$ such as

$$W_{\text{out}} = R^{-1}Q^T Y. \tag{D.7}$$

One advantage of using QR is that it avoids forming the Gram matrix $A^T A$ of the normal equations which can be singular.

## D.3 Computational Complexity

One special outcome of the formalism presented in Chapter 5, section 4, is that it separates the homogeneous part of the differential equation from the initial conditions and forces. In other words, if we investigate the normal equations for $W_{\text{out}}$, we see that

$$W_{\text{out}} = (\hat{D}_H^T \hat{D}_H + \bar{D}_H^T \bar{D}_H)^{-1}(D_H^T f'(t) + \bar{D}_H^T u'_{\text{ic}}). \tag{D.8}$$

Notice that the forces and initial conditions appear outside the matrix inversion. This is a particularly important feature as it allows us to scale rapidly when the differential equation is fixed and the solution to many initial conditions or forces is required. In fact, the computational complexity of the inversion is $\mathcal{O}(h^3)$ where $h$ is the number of output neurons of the final hidden layer and the multiplication is $\mathcal{O}(h^2 m)$, where m is the number of initial conditions and forces. Therefore, if the differential equation is fixed the total computational cost of computing $W_{\text{out}}$ is $\mathcal{O}(h^3 + mh^2)$. However, if the differential equation is not fixed and varies across all m samples, then the inversion has to be computed $m$ times such that the total computational cost is $\mathcal{O}(mh^3 + mh^2)$.

## D.4 Training Bundles

Typically, the more training bundles we use, the more diverse the hidden states have to be in order to generate accurate solutions for all the training differential equations. As such, we would expect that a diverse set of hidden states should also result in better inference for unseen differential equations. Experimentally, we find and show in Fig. D.1 that for first order differential equations this is true:

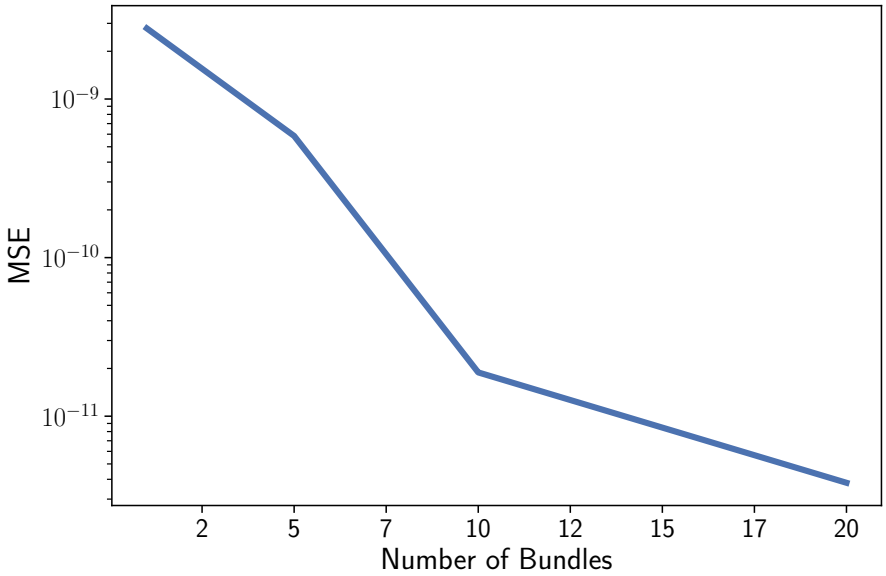



**Figure D.1:** Test MSE as a function of number of bundles. The more equations (bundles) we simultaneously train, the better our test accuracy gets.

Of course doing this also increases the training overhead, thus empirically we use 10 equations consistently across our experiments of ordinary differential equations.

# References


Rhone, Trevor David et al. (2020). "Data-driven studies of magnetic two-dimensional materials". In: *Scientific Reports* 10.1, p. 15795.

Melchionna, Simone et al. (2011). "Endothelial shear stress from large-scale blood flow simulations". In: *Philosophical Transactions of the Royal Society A* 369, pp. 2354–2361.

Achdou, Yves et al. (2014). "Partial differential equation models in macroeconomics". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 372.2028, p. 20130397.

Sanchez-Gonzalez, Alvaro, Jonathan Godwin, et al. (2020). "Learning to Simulate Complex Physics with Graph Networks". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Proceedings of Machine Learning Research 119, pp. 8459–8468.

Karniadakis, George Em et al. (June 2021). "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6, pp. 422–440.

Li, Ziyao et al. (2021). "Conformation-Guided Molecular Representation with Hamiltonian Neural Networks". In: *International Conference on Learning Representations*.

He, Kaiming, Georgia Gkioxari, et al. (2017). "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988.

Devlin, Jacob et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein et al., pp. 4171–4186.

Bapst, V. et al. (Apr. 2020). "Unveiling the predictive power of static structure in glassy systems". In: *Nature Physics* 16.4. Number: 4 Publisher: Nature Publishing Group, pp. 448–454.

Rath, Katharina et al. (May 2021). "Symplectic Gaussian Process Regression of Hamiltonian Flow Maps". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.5. arXiv: 2009.05569, p. 053121. (Visited on 03/27/2022).

Dunlop, Matthew M. et al. (2018). "How Deep Are Deep Gaussian Processes?" In: *Journal of Machine Learning Research* 19.54, pp. 1–46. URL: `http://jmlr.org/papers/v19/18-015.html`.

Marsden, J. E. et al. (May 2001). "Discrete mechanics and variational integrators". In: *Acta Numerica* 10, pp. 357–514.

Lagrange, Joseph-Louis (1811). "Mécanique Analytique". In: *Ve Courcier*.

Hornik, Kurt et al. (Jan. 1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366.

He, Kaiming, Xiangyu Zhang, et al. (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Goodfellow, Ian et al. (2016). *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press.

Kingma, Diederik P. et al. (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio et al.

Maclaurin, Dougal et al. (2015). "Autograd: Effortless Gradients in Numpy". In: *ICML 2015 AutoML Workshop*.

Rumelhart, David E. et al. (1986). "Learning internal representations by error propagation". In:

Chen, Ricky T. Q., Yulia Rubanova, et al. (2018). "Neural Ordinary Differential Equations". In: ed. by S. Bengio et al. Curran Associates, Inc., pp. 6571–6583.

Hochreiter, Sepp et al. (Nov. 1997). "Long Short-Term Memory". In: *Neural Comput.* 9.8, pp. 1735–1780.

Battaglia, Peter W., Jessica B. Hamrick, et al. (Oct. 2018). "Relational inductive biases, deep learning, and graph networks". In: *arXiv:1806.01261 [cs, stat]*. arXiv: 1806.01261.

Rupp, Matthias et al. (Jan. 2012). "Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning". In: *Physical Review Letters* 108.5. Publisher: American Physical Society, p. 058301.

Smith, J. S. et al. (2017). "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost". In: *Chemical Science* 8.4, pp. 3192–3203.

Lagaris, I. E. et al. (Sept. 1998). "Artificial Neural Networks for Solving Ordinary and Partial Differential Equations". In: *IEEE Transactions on Neural Networks* 9.5. arXiv: physics/9705023, pp. 987–1000.

Howse, James W. et al. (1996). "Gradient and Hamiltonian Dynamics Applied to Learning in Neural Networks". In: ed. by D. S. Touretzky et al. MIT Press, pp. 274–280.

Raissi, Maziar et al. (Nov. 2017). "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations". In: *arXiv:1711.10561 [cs, math, stat]*. arXiv: 1711.10561.

Witkoskie, James B. et al. (Jan. 2005). "Neural Network Models of Potential Energy Surfaces: Prototypical Examples". In: *Journal of Chemical Theory and Computation* 1.1, pp. 14–23.

Pukrittayakamee, A. et al. (Apr. 2009). "Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks". In: *The Journal of Chemical Physics* 130.13, p. 134101.

Yao, Kun et al. (2018). "The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics". In: *Chemical Science* 9.8, pp. 2261–2269.

Battaglia, Peter W., Razvan Pascanu, et al. (Dec. 2016). "Interaction Networks for Learning about Objects, Relations and Physics". In: *arXiv:1612.00222 [cs]*. arXiv: 1612.00222.

Sanchez-Gonzalez, Alvaro, Nicolas Heess, et al. (July 2018). "Graph Networks as Learnable Physics Engines for Inference and Control". In: *Proceedings of the 35th International Conference on Machine Learning*. Proceedings of Machine Learning Research 80. Ed. by Jennifer Dy et al., pp. 4470–4479.

Seo, Sungyong and Yan Liu (Feb. 2019). "Differentiable Physics-informed Graph Networks". In: *arXiv:1902.02950 [cs, stat]*. arXiv: 1902.02950.

Cranmer, Miles, Alvaro Sanchez Gonzalez, et al. (2020). "Discovering Symbolic Models from Deep Learning with Inductive Biases". In: *Advances in Neural Information Processing Systems* 33. Ed. by H. Larochelle et al., pp. 17429–17442.

Seo, Sungyong, Chuizheng Meng, et al. (2020). "Physics-aware Difference Graph Networks for Sparsely-Observed Dynamics". In: *International Conference on Learning Representations*.

Lamb, Luis et al. (Mar. 2020). "Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective". In: *arXiv:2003.00330 [cs]*. arXiv: 2003.00330.

Cranmer, Miles, Sam Greydanus, et al. (Mar. 2020). "Lagrangian Neural Networks". In: *arXiv:2003.04630 [physics, stat]*. arXiv: 2003.04630.

Sanchez-Gonzalez, Alvaro, Victor Bapst, et al. (Sept. 2019). "Hamiltonian Graph Networks with ODE Integrators". In: *arXiv:1909.12790 [physics]*. arXiv: 1909.12790.

He, Kaiming, Xiangyu Zhang, et al. (2015). "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385.

Kobyzev, Ivan et al. (2020). "Normalizing Flows: An Introduction and Review of Current Methods". In: *PAMI*. arXiv: `1908.09257 [stat.ML]`.

Lu, Yiping et al. (2017). "Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations". In: *CoRR* abs/1710.10121. arXiv: `1710.10121`.

Chen, Ricky T. Q., Brandon Amos, et al. (2021). "Learning Neural Event Functions for Ordinary Differential Equations". In: *International Conference on Learning Representations*.

Zhong, Yaofeng Desmond and Naomi Leonard (2020). "Unsupervised Learning of Lagrangian Dynamics from Images for Prediction and Control". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 10741–10752.

Zhong, Yaofeng Desmond, Biswadip Dey, et al. (2020a). "Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control". In: *International Conference on Learning Representations*.

Saemundsson, Steindor et al. (June 2020). "Variational Integrator Networks for Physically Structured Embeddings". In: ISSN: 2640-3498. PMLR, pp. 3078–3087.

Dupont, Emilien et al. (2019). "Augmented Neural ODEs". In: *Advances in Neural Information Processing Systems* 32. Ed. by H. Wallach et al.

Zhang, Ruiyang et al. (Sept. 2020). "Physics-Informed Multi-LSTM Networks for Metamodeling of Nonlinear Structures". In: *Computer Methods in Applied Mechanics and Engineering* 369. arXiv: 2002.10253, p. 113226.

Wang, Hengjie et al. (Apr. 2021). "Train Once and Use Forever: Solving Boundary Value Problems in Unseen Domains with Pre-trained Deep Learning Models". In: *arXiv:2104.10873 [physics]*. arXiv: 2104.10873.

Wang, Sifan and Paris Perdikaris (Mar. 2021). "Deep learning of free boundary and Stefan problems". In: *Journal of Computational Physics* 428. arXiv: 2006.05311, p. 109914.

Zhai, Hanfeng et al. (May 2021a). "Inferring micro-bubble dynamics with physics-informed deep learning". In: *arXiv:2105.07179 [physics]*. arXiv: 2105.07179.

Sahli Costabal, Francisco et al. (2020). "Physics-Informed Neural Networks for Cardiac Activation Mapping". In: *Frontiers in Physics* 8. Publisher: Frontiers.

Wolff, Taco de et al. (June 2021). "Towards Optimally Weighted Physics-Informed Neural Networks in Ocean Modelling". In: *arXiv:2106.08747 [physics]*. arXiv: 2106.08747.

Flamant, Cedric et al. (June 2020). "Solving Differential Equations Using Neural Network Solution Bundles". In: *arXiv:2006.14372 [physics]*. arXiv: 2006.14372.

Yang, Liu et al. (Nov. 2018). "Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations". In: *arXiv:1811.02033 [cs, math, stat]*. arXiv: 1811.02033.

Sirignano, Justin et al. (Dec. 2018). "DGM: A deep learning algorithm for solving partial differential equations". In: *Journal of Computational Physics* 375. arXiv: 1708.07469, pp. 1339–1364.

Raissi, M. et al. (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.

Greydanus, Samuel et al. (2019). "Hamiltonian Neural Networks". In: ed. by H. Wallach et al. Curran Associates, Inc., pp. 15379–15389.

Lew, Adrian et al. (2004). "An overview of variational integrators". In: *Finite Element Methods*, p. 18.

Lutter, Michael et al. (2019). "Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning". In: *International Conference on Learning Representations*.

Zhu, Aiqing et al. (Apr. 2020). "Deep Hamiltonian networks based on symplectic integrators". In: *arXiv:2004.13830 [cs, math]*. arXiv: 2004.13830.

Chen, Zhengdao et al. (2020). "Symplectic Recurrent Neural Networks". In: *International Conference on Learning Representations*.

Jin, Pengzhan et al. (Dec. 2020). "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems". In: *Neural Networks* 132, pp. 166–179.

Geist, A. Rene et al. (Apr. 2020). "Learning Constrained Dynamics with Gauss Principle adhering Gaussian Processes". In: *arXiv:2004.11238 [cs, eess, stat]*. arXiv: 2004.11238.

Atkinson, Steven (June 2020). "Bayesian Hidden Physics Models: Uncertainty Quantification for Discovery of Nonlinear Partial Differential Operators from Data". In: *arXiv:2006.04228 [cs, stat]*. arXiv: 2006.04228.

Iten, Raban et al. (Jan. 2020). "Discovering physical concepts with neural networks". In: *Physical Review Letters* 124.1. arXiv: 1807.10300, p. 010508.

Udrescu, Silviu-Marian et al. (Apr. 2021). "Symbolic pregression: Discovering physical laws from distorted video". In: *Phys. Rev. E* 103 (4), p. 043307.

Barmparis, G.D. et al. (2020). "Robust prediction of complex spatiotemporal states through machine learning with sparse sensing". In: *Physics Letters A* 384.15, p. 126300.

Zhai, Hanfeng et al. (May 2021b). "Inferring micro-bubble dynamics with physics-informed deep learning". In: *arXiv:2105.07179 [physics]*. arXiv: 2105.07179.

Chang, Bo et al. (Nov. 2017). "Reversible Architectures for Arbitrarily Deep Residual Neural Networks". In: *arXiv:1709.03698 [cs, stat]*. arXiv: 1709.03698.

Desai, Shaan A., Marios Mattheakis, David Sondak, et al. (Sept. 2021). "Port-Hamiltonian neural networks for learning explicit time-dependent dynamical systems". In: *Phys. Rev. E* 104 (3), p. 034312.

Yu, Haijun et al. (Oct. 2020). "OnsagerNet: Learning Stable and Interpretable Dynamics using a Generalized Onsager Principle". In: *arXiv:2009.02327 [physics]*. arXiv: 2009.02327.

Hills, Daniel J.A. et al. (Nov. 2015). "An algorithm for discovering Lagrangians automatically from data". In: *PeerJ Computer Science* 1, e31.

Schmidt, Michael et al. (2009). "Distilling Free-Form Natural Laws from Experimental Data". In: *Science* 324.5923. Publisher: American Association for the Advancement of Science _eprint: https://science.sciencemag.org/content/324/5923/81.full.pdf, pp. 81–85.

Silva, Brian M. de et al. (2020). "Discovery of Physics From Data: Universal Laws and Discrepancies". In: *Frontiers in Artificial Intelligence* 3, p. 25.

Toth, Peter et al. (2020). "Hamiltonian Generative Networks". In: *International Conference on Learning Representations*.

Choudhary, Anshul et al. (June 2020). "Physics-enhanced neural networks learn order and chaos". In: *Phys. Rev. E* 101 (6), p. 062207.

Finzi, Marc, Ke Alexander Wang, et al. (2020). "Simplifying Hamiltonian and Lagrangian Neural Networks via Explicit Constraints". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 13880–13889.

Mattheakis, Marios, David Sondak, et al. (Feb. 2020). "Hamiltonian Neural Networks for solving differential equations". In: *arXiv:2001.11107 [physics]*. arXiv: 2001.11107.

Toussaint, Marc et al. (June 2018). "Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning". In: Robotics: Science and Systems Foundation.

Schaft, Arjan van der (May 2007). "Port-Hamiltonian systems: an introductory survey". In: ed. by Marta Sanz-Solé et al. Zuerich, Switzerland: European Mathematical Society Publishing House, pp. 1339–1365.

Ortega, Romeo et al. (2002). "Interconnection and damping assignment passivity-based control of port-controlled Hamiltonian systems". In: *Automatica* 38.4, pp. 585–596.

Acosta, J. A. et al. (Dec. 2005). "Interconnection and damping assignment passivity-based control of mechanical systems with underactuation degree one". In: *IEEE Transactions on Automatic Control* 50.12. Conference Name: IEEE Transactions on Automatic Control, pp. 1936–1955.

Zheng, Min et al. (Nov. 2018). *Time-Varying Impedance Control of Port Hamiltonian System with a New Energy-Storing Tank*. Research Article. ISSN: 1076-2787 Pages: e8134230 Publisher: Hindawi Volume: 2018.

Cherifi, Karim (Oct. 2020). "An overview on recent machine learning techniques for Port Hamiltonian systems". In: *Physica D: Nonlinear Phenomena* 411, p. 132620.

Zhong, Yaofeng Desmond, Biswadip Dey, et al. (Apr. 2020b). "Dissipative SymODEN: Encoding Hamiltonian Dynamics with Dissipation and Control into Deep Learning". In: *arXiv:2002.08860 [cs, eess, stat]*. arXiv: 2002.08860.

Finzi, Marc, Samuel Stanton, et al. (May 2020). "Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data". In: *arXiv:2002.12880 [cs, stat]*. arXiv: 2002.12880.

Kaxiras, Efthimios et al. (Apr. 2020). "The first 100 days: modeling the evolution of the COVID-19 pandemic". In: *arXiv:2004.14664 [q-bio]*. arXiv: 2004.14664.

Dormand, J. R. et al. (1987). "Families of Runge-Kutta-Nystrom Formulae". In: *IMA Journal of Numerical Analysis* 7.2, pp. 235–250.

Mattheakis, Marios, Hayden Joy, et al. (2021). "Unsupervised Reservoir Computing for Solving Ordinary Differential Equations". In: *arXiv:2108.11417 [cs:LG]*. arXiv: 2108.11417.

McClenny, Levi et al. (Sept. 2020). "Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism". In: *arXiv:2009.04544 [cs, stat]*. arXiv: 2009.04544.

Guo, Yali et al. (Jan. 2021). "Transfer learning of chaotic systems". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.1. arXiv: 2011.09970, p. 011104.

Desai, Shaan A., Marios Mattheakis, and Stephen J. Roberts (Sept. 2021). "Variational integrator graph networks for learning energy-conserving dynamical systems". In: *Phys. Rev. E* 104 (3), p. 035310.

Yin, Yuan et al. (2021). "Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting". In: *International Conference on Learning Representations*.

Desai, Shaan et al. (2021). "Port-Hamiltonian Neural Networks for Learning Explicit Time-Dependent Dynamical Systems". In: *CoRR* abs/2107.08024. _eprint: 2107.08024.

Lee, Kookjin et al. (June 2021). "Machine learning structure preserving brackets for forecasting irreversible processes". In: *arXiv:2106.12619 [physics]*. arXiv: 2106.12619.

Hochlehnert, Andreas et al. (2021). "Learning Contact Dynamics using Physically Structured Neural Networks". In: *AISTATS*, pp. 2152–2160.

Chen, Feiyu et al. (2020). "NeuroDiffEq: A Python package for solving differential equations with neural networks". In: *Journal of Open Source Software* 5.46, p. 1931.

Lu, Lu et al. (2021). "A deep learning library for solving differential equations". In: *SIAM Review* 63, pp. 208–228.

Hennigh, Oliver et al. (2020). "NVIDIA SimNet: an AI-accelerated multi-physics simulation framework". In: *arXiv:2012.07938 [physics.flu-dyn]*.

Jaeger, Herbert et al. (2004). "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication". In: *Science* 304.5667, pp. 78–80.

Xu, Zhi-Qin John et al. (Oct. 2019). "Training behavior of deep neural network in frequency domain". In: *arXiv:1807.01251 [cs, math, stat]*. arXiv: 1807.01251.

Schwartz, Matthew (2017). "Lecture 3: Coupled oscillators". In: p. 6.

Wang, Sifan, Shyam Sankaran, et al. (Mar. 2022). "Respecting causality is all you need for training physics-informed neural networks". In: *arXiv:2203.07404 [nlin, physics:physics, stat]*. arXiv: 2203.07404.

Lemos, Pablo et al. (Feb. 2022). "Rediscovering orbital mechanics with machine learning". In: *arXiv:2202.02306 [astro-ph]*. arXiv: 2202.02306. (Visited on 03/27/2022).

Aitio, Antti et al. (Dec. 2021). "Predicting battery end of life from solar off-grid system field data using machine learning". In: *Joule* 5.12. arXiv: 2107.13856. (Visited on 03/27/2022).

Li, Weihan et al. (2021). "One-shot battery degradation trajectory prediction with deep learning". In: *Journal of Power Sources* 506, p. 230024.

Grimm, Viktor et al. (2021). "Estimating the time-dependent contact rate of SIR and SEIR models in mathematical epidemiology using physics-informed neural networks". In: *ETNA - Electronic Transactions on Numerical Analysis* 56, pp. 1–27. (Visited on 03/27/2022).

Geneva, Nicholas et al. (June 2021). "Transformers for Modeling Physical Systems". In: *arXiv:2010.03957 [physics]*. arXiv: 2010.03957.

Li, Bei et al. (Apr. 2021). "ODE Transformer: An Ordinary Differential Equation-Inspired Model for Neural Machine Translation". In: *arXiv:2104.02308 [cs]*. arXiv: 2104.02308.

Runge, C. (1895). "Ueber die numerische Auflösung von Differentialgleichungen". In: *Mathematische Annalen* 46.2, pp. 167–178.

*References*

Butcher, John (2008). "Runge–Kutta Methods". In: *Numerical Methods for Ordinary Differential Equations*. John Wiley and Sons, Ltd. Chap. 3, pp. 137–316.

Yoshida, Haruo (1990). "Construction of higher order symplectic integrators". In: *Physics Letters A* 150.5, pp. 262–268.

Forest, Etienne et al. (1990). "Fourth-order symplectic integration". In: *Physica D: Nonlinear Phenomena* 43.1, pp. 105–117.