B.Comp. Dissertation

Physics-informed Machine Learning

Measuring Chaotic Behaviour with Hamiltonian Neural Networks

by

Wilson Widyadhana

Department of Computer Science

School of Computing

National University of Singapore

2024/2025

B.Comp. Dissertation

Physics-informed Machine Learning
Measuring Chaotic Behaviour with Hamiltonian Neural Networks

by

Wilson Widyadhana

Department of Computer Science

School of Computing

National University of Singapore

2024/2025

Project No: H0811740
Advisor: Dr. Eric Han Liang Wee, Assoc. Prof. Stéphane Bressan
Main Evaluator: Asst. Prof. Shao Lin
Deliverables:
Report: 1 Volume

Acknowledgement

I would like to first thank my friends, family, advisors, and mentors. Moreover, I would like to specifically thank Dr. Eric Han Liang Wee, Prof. Stéphane Bressan, Prof. Gabriel Lemarie, Khoo Zi-Yu, and Dawen Wu for their invaluable advice and mentorship throughout the duration of the project. Without their continued support and encouragement, I would not have been able to complete this project.

List of Figures

Table of Contents

Abstract

Physics-informed machine learning has led to important discoveries in the intersection of physics and computer science. Particularly interesting is the topic of chaotic systems and their modelling using neural networks. Such machine learning methods would have the ability to be deployed with fewer computational resources compared to classical methods that use trajectory evolution.

We propose that the quality of predictions given by several performance metrics relates to the degree of chaotic behaviour present in different regions of the system. We show that the proposed statement is true across different systems and with various models. We also show that the quality of predictions across time-steps relates well to the measure of chaos as described above.

Subject Descriptors:
Artificial Intelligence
Machine Learning
Neural Networks
Supervised Learning by Regression

Keywords:
Physics, artificial intelligence, machine learning, neural networks, Hamiltonian

Implementation Software and Hardware:
Python, PyTorch, JAX, Ray Tune, SoC Compute Cluster

Chapter 1

Introduction

1.1 Motivation

Artificial intelligence and machine learning have transformed many fields. A few of these advancements include assisting researchers in making predictions on the shape and properties of proteins (Jumper et al., 2021) and having chatbots that can communicate on a human-like level (OpenAI et al., 2024). This is not limited to only such fields, however, as deep neural networks (DNNs) have also been applied to physics with much success.

Physics-informed neural networks (PINNs) have been used in a variety of situations, such as in the modelling of dynamical systems (Greydanus, Dzamba, & Yosinski, 2019) or in the discovery of partial differential equations from data (Raissi, Perdikaris, & Karniadakis, 2019) (Raissi & Karniadakis, 2018). They often incorporate physical laws into the architecture, data, or training (Khoo, Wu, Low, & Bressan, 2024) which can improve the ability of such models to perform on tasks that are difficult to perform with standard neural networks.

Chaotic dynamics are a natural part of the world around us. They are found in a variety of different fields, including meteorology (Buizza, 2002) and robotics (Zang, Iqbal, Zhu, Liu, & Zhao, 2016). A common theme across all of these is that such dynamics are very sensitive to initial conditions, which leads to difficulty in long-term predictions of the eventual state of the system. That being said, one usually has a good understanding of the equations modelling the changes in the system. These are often incorporated into a single conserved quantity known as

the Hamiltonian.

Recent advances in machine learning have led to the invention of architectures that aim to conserve a similar quantity (Greydanus et al., 2019). This is primarily done by modifying the loss function used to train the models to constrain the predictions to those that meet the above criteria. By doing this, one is able to integrate physical laws – such as conservation of the Hamiltonian – directly into the models, which is important if we are to have accurate predictions within dynamical systems.

In this project, we would like to explore the possibility of predicting the degree of chaotic behaviour inside dynamical systems using one of the aforementioned recent architectures known as the Hamiltonian neural network (HNN) (Greydanus et al., 2019). This is crucial as detecting chaotic behaviour often consists of evolving the system over long time-steps using symplectic integrators (Hairer, 2010) and then computing the local growth rate or the Lyapunov exponent (Goldstein, Poole, & Safko, 2002) (Datseris, 2018). Such computations – which are often non-parallelisable, require complex algorithms, and necessitate large amounts of data to be used for a single prediction – take up valuable computational resources and time. We shall focus on additive separable systems in this project. Nevertheless, we believe that the methods employed here can be adapted to non-additive separable systems.

We believe that there is a relationship between a model's predictive ability and the degree of chaotic behaviour present in regions of the system. In order to determine the extent to which this is true, we shall use different kinds of models and metrics before analysing them in detail.

1.2 Overview

This report contains several chapters that cover the motivation, background, related work, methodology, results, and conclusion of this project. We have covered the first in this chapter. Chapter 2 serves as a general introduction to the necessary background used in the later chapters. Afterwards, Chapter 3 expands on this to the specific research that we have built upon for this project. Chapter 4 deliberates on the work we have done and provides support to our experiments. Chapter 5 will expand on the details of the experiments we have conducted, as

well as provide discussions with respect to results. Finally, Chapter 6 shall give the reader a brief summary of experimental results and possible directions for future work.

Chapter 2

Background

2.1 Lagrangian and Hamiltonian Mechanics

In order to understand Hamiltonian neural networks and other architectures based on the Hamiltonian, it is imperative that we first comprehend the theory behind both the Lagrangian and the Hamiltonian. We shall start with Lagrangian mechanics, before using the Legendre transform to arrive at Hamiltonian mechanics.

2.1.1 Lagrangian Mechanics

Lagrangian mechanics is a specific way of describing dynamical systems via the use of the Lagrangian, a quantity conserved within many dynamical systems. We shall start by deriving the Euler-Lagrange equations before moving on to the Lagrange equations of motion (Goldstein et al., 2002) (Lemarié, 2022).

Define the integral (Goldstein et al., 2002) (Lemarié, 2022):

$$S(y) := \int_{x_1}^{x_2} f(y(x), y'(x), x)dx \tag{2.1}$$

where $f(y(x), y'(x), x)$ is a functional (i.e., a function that takes in other functions as inputs), and $y(x) = y_0(x) + \varepsilon\eta(x)$ such that $\eta(x_1) = \eta(x_2) = 0$ and $\varepsilon \ll 1$. We can imagine $y(x)$ to be some function dependent on a fixed trajectory $y_0(x)$ that is perturbed by some $\varepsilon\eta(x)$. The

integral is hence the limit of the Riemann sum of some function based on such a trajectory.

Using the above integral, we can compute $\frac{dS(y)}{d\varepsilon}$ (Goldstein et al., 2002) (Lemarié, 2022):

$$\frac{dS(y)}{d\varepsilon} = \frac{d}{d\varepsilon}\int_{x_1}^{x_2} f(y_0(x)+\varepsilon\eta(x), y_0'(x)+\varepsilon\eta'(x), x)dx \tag{2.2}$$

$$= \int_{x_1}^{x_2}\left(\frac{\partial f}{\partial y}\frac{\partial y}{\partial \varepsilon}+\frac{\partial f}{\partial y'}\frac{\partial y'}{\partial \varepsilon}\right)dx \tag{2.3}$$

$$= \int_{x_1}^{x_2}\left(\frac{\partial f}{\partial y}\eta(x)+\frac{\partial f}{\partial y'}\eta'(x)\right)dx \tag{2.4}$$

$$= \int_{x_1}^{x_2}\left(\frac{\partial f}{\partial y}\eta(x)-\eta(x)\frac{d}{dx}\frac{\partial f}{\partial y'}\right)dx + \underbrace{\frac{\partial f}{\partial y'}\eta(x)\Big|_{x_1}^{x_2}}_{0} \quad \text{(integrating by parts)} \tag{2.5}$$

$$= \int_{x_1}^{x_2}\eta(x)\left(\frac{\partial f}{\partial y}-\frac{d}{dx}\frac{\partial f}{\partial y'}\right)dx \tag{2.6}$$

$$= 0 \quad \text{(minimising the derivative)} \tag{2.7}$$

The above leads directly to the Euler-Lagrange equation (Goldstein et al., 2002) (Lemarié, 2022):

Theorem 2.1.1 (Euler-Lagrange equation (Goldstein et al., 2002) (Lemarié, 2022))**.** *Given $S(y) = \int_{x_1}^{x_2} f(y(x), y'(x), x)dx$, we have that the function $y(x)$ which minimises $S(y)$ fulfils the following condition:*

$$\frac{\partial f}{\partial y}-\frac{d}{dx}\frac{\partial f}{\partial y'} = 0 \tag{2.8}$$

Theorem 2.1.1 can be used to solve a few mathematical problems, most famous of which is the tautochrone problem (Weisstein, 2025). We restate the aforementioned theorem below in more general terms, to allow for its application in multiple dimensions.

Corollary 2.1.1.1 (Euler-Lagrange equations with indices (Goldstein et al., 2002))**.** *Theorem 2.1.1 implies that for $S(\{y_i\}_{i=1}^n) = \int_{x_1}^{x_2} f(\{y_i\}_{i=1}^n, \{y_i'\}_{i=1}^n, x)dx$, we have:*

$$\frac{\partial f}{\partial y_i}-\frac{d}{dx}\frac{\partial f}{\partial y_i'} = 0, \ \ \forall i = 1, 2, ..., n \tag{2.9}$$

Corollary 2.1.1.1, however, is not sufficient to describe dynamical systems. In order to do so, we will need to define the **Lagrangian** and the **action** (Goldstein et al., 2002).

Definition 2.1.2 (Lagrangian)**.** The Lagrangian, $\mathcal{L}$ is defined as (Goldstein et al., 2002):

$$\mathcal{L} := T - V \tag{2.10}$$

where T is the kinetic energy and V is the potential energy of a system.

Definition 2.1.3 (Action)**.** The action, $S(\mathcal{L})$, is defined as (Goldstein et al., 2002):

$$S(\mathcal{L}) := \int_{t_1}^{t_2} \mathcal{L}(q(t), \dot{q}(t), t)dt \tag{2.11}$$

Definition 2.1.4 (Principle of least action (Goldstein et al., 2002))**.** The principle of least action attempts to minimise the action throughout a trajectory between two points.

In many systems, we have that the principle of least action is obeyed. This implies that we are able to use Corollary 2.1.1.1 with some variable substitutions in order to arrive at the Lagrange equations of motion below.

Theorem 2.1.5 (Lagrange equations of motion (Goldstein et al., 2002))**.** *From Corollary 2.1.1.1, we can substitute y_i for q_i, x for t, and f for $\mathcal{L}$ in order to arrive at the Lagrange equations for motion:*

$$\frac{\partial \mathcal{L}}{\partial q_i} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} = 0, \ \ \forall i = 1, 2, ..., n \tag{2.12}$$

The above equation is often used as a basis to analytically solve for equations of motion of various dynamical systems (Goldstein et al., 2002). We shall also describe, in Chapter 3, how this equation has been used by several authors to create different types of physics-informed neural networks.

2.1.2 Hamiltonian Mechanics

Before deriving Hamilton's equations of motion, we will have to state the Legendre transform without proof (Cline, 2021) (Goldstein et al., 2002). This transform allows one to change the formulation of various equations via the relationship of its components.

Theorem 2.1.6 (Legendre transform (Cline, 2021) (Goldstein et al., 2002))**.** *The Legendre transform of a function $F(\boldsymbol{u}, \boldsymbol{w})$ and $G(\boldsymbol{v}, \boldsymbol{w})$, where $\boldsymbol{v} = \nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{w})$ is such that $\boldsymbol{u} = \nabla_{\boldsymbol{v}} G(\boldsymbol{v}, \boldsymbol{w})$. Furthermore, $G(\boldsymbol{v}, \boldsymbol{w}) + F(\boldsymbol{u}, \boldsymbol{w}) = \boldsymbol{u} \cdot \boldsymbol{v}$.*

Using the Lagrange equations of motion found in Theorem 2.1.5 as well as the Legendre transform in Theorem 2.1.6, we find the equation below (Goldstein et al., 2002):

Theorem 2.1.7 (Hamilton's equations of motion (Goldstein et al., 2002))**.** *Hamilton's equations of motion, where q_i and p_i are conjugate position and momenta respectively, can be written as (Goldstein et al., 2002):*

$$\mathcal{H}(\{q_i\}_{i=1}^n, \{p_i\}_{i=1}^n, t) = \sum_{i=1}^n p_i \dot{q}_i - \mathcal{L}(\{q_i\}_{i=1}^n, \{\dot{q}_i\}_{i=1}^n, t) \tag{2.13}$$

where $\mathcal{L}(\{q_i\}_{i=1}^n, \{\dot{q}_i\}_{i=1}^n, t)$ is the Lagrangian.

In constructing the proof, we will take the notation $\mathbf{q}$ to be equivalent to $\{q_i\}_{i=1}^n$. The same will apply for $\dot{\mathbf{q}}$, $\mathbf{p}$, and $\dot{\mathbf{p}}$.

Proof. (Goldstein et al., 2002) We have that $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t)$ and $\mathcal{H}(\mathbf{q}, \mathbf{p}, t)$ are linked by the equation $\mathbf{p} = \nabla_{\dot{\mathbf{q}}} \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t)$. Thus by the Legendre transform in Theorem 2.1.6 we have that $\dot{\mathbf{q}} = \nabla_{\mathbf{p}} \mathcal{H}(\mathbf{q}, \mathbf{p}, t)$ and $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t) + \mathcal{H}(\mathbf{q}, \mathbf{p}, t) = \mathbf{p} \cdot \dot{\mathbf{q}}$ as desired. ∎

Also by the Legendre transform, we have the following:

$$\frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial \mathbf{p}} = \dot{\mathbf{q}} \tag{2.14}$$

$$\frac{\partial \mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial \mathbf{q}} = -\dot{\mathbf{p}} \tag{2.15}$$

Hence, given the Hamiltonian of a system, we will be able to derive its equations of motion. This will then allow us to evolve the phase-space representation of the system across time. Following Khoo et al. (2024), we can define Hamilton's equations in a more succinct manner below.

Let $\mathbf{z} = \begin{pmatrix} \mathbf{q} \\ \mathbf{p} \end{pmatrix} \in \mathbb{R}^{2\times n}$. This means that we can now write the Hamiltonian as $\mathcal{H}(\mathbf{z}) : \mathbb{R}^{2\times n} \to \mathbb{R}$. We then have (Khoo et al., 2024):

$$\dot{\mathbf{z}} = \begin{pmatrix} \frac{\partial \mathcal{H}(\mathbf{z})}{\partial \mathbf{p}} \\ -\frac{\partial \mathcal{H}(\mathbf{z})}{\partial \mathbf{q}} \end{pmatrix} = \mathbf{J}\nabla_{\mathbf{z}}\mathcal{H}(\mathbf{z}) \tag{2.16}$$

$$\text{where } \mathbf{J} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{\text{-}I} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{2n\times 2n}, \mathbf{I} \in \mathbb{R}^{n\times n}, \mathbf{0} \in \mathbb{R}^{n\times n} \tag{2.17}$$

In this project, we focus on systems that are additive separable. Our definition of this will follow Khoo et al. (2024):

Definition 2.1.8 (Additive Separability (Khoo et al., 2024))**.** An additive separable system is defined as where the Hamiltonian can be written as follows:

$$\mathcal{H}(\mathbf{q}, \mathbf{p}) = f(\mathbf{q}) + g(\mathbf{p}) \tag{2.18}$$

where $f(\mathbf{q})$ and $g(\mathbf{p})$ are arbitrary functions.

A few examples are as follows:

Example 2.1.9 (Hénon-Heiles System)**.** The Hamiltonian of the (standard) Hénon-Heiles system is defined as:

$$\mathcal{H}(q_1, q_2, p_1, p_2) = \frac{1}{2}(p_1^2 + p_2^2) + \frac{1}{2}\left(q_1^2 + q_2^2\right) + \left(q_1^2 q_2 - \frac{q_2^3}{3}\right) \tag{2.19}$$

Define $f(\mathbf{q}) = \frac{1}{2}\left(q_1^2 + q_2^2\right) + \left(q_1^2 q_2 - \frac{q_2^3}{3}\right)$ and $g(\mathbf{p}) = \frac{1}{2}\left(p_1^2 + p_2^2\right)$. Then $\mathcal{H}(\mathbf{q}, \mathbf{p}) = f(\mathbf{q}) + g(\mathbf{p})$.

.

Example 2.1.10 (Non-additive Separable System)**.** The Hamiltonian of an arbitrary system is defined as:

$$\mathcal{H}(q_1, q_2, p_1, p_2) = \sum_{i=1}\sum_{j\neq i} q_i p_j \tag{2.20}$$

As we are unable to find any $\mathcal{H}(\mathbf{q}, \mathbf{p}) = f(\mathbf{q}) + g(\mathbf{p})$, this system is non-additive separable.

.

2.2 Chaos and Chaotic Behaviour

2.2.1 Rigorous Definition of Chaos

Chaos does not have a standard definition in most of the literature. Most mathematically-inclined texts use the more rigorous definition mentioned in Hasselblatt and Katok (2003) and given below:

Definition 2.2.1 (Properties of chaotic systems (Hasselblatt & Katok, 2003))**.** A chaotic system is defined as any system with these properties:

1. Sensitive to initial conditions;
2. Topologically transitive; and
3. Have dense periodic orbits.

Using the three conditions defined above can be difficult, as evaluating each of them can be computationally expensive or be quantitatively impractical to use. Moreover, we are specifically focusing on detecting and measuring the degree of chaos in chaotic **regions** instead of entire systems.

Hence, we will be mostly using the two definitions of chaotic behaviour found below as both necessary and sufficient conditions for determining the presence of chaos. The former, namely the use of Poincaré surface of sections, requires visualisations of the trajectories; the latter, local growth rates, can be applied numerically.

2.2.2 Poincaré Surface of Sections

We are able to detect chaos by the presence of hyperbolic points in phase-space. Hyperbolic points exist in between elliptic fixed points (Goldstein et al., 2002). A fixed point is a point in phase-space where $\mathbf{z}(t) = \mathbf{z}(0)$ for all t (Goldstein et al., 2002). Elliptic fixed points are visible in Poincaré surface of section as being surrounded with elliptical orbits (Goldstein et al., 2002). Hyperbolic points, in comparison, can be seen as the crossing points of orbits that do not surround elliptic fixed points (Goldstein et al., 2002).

These are visible by the use of Poincaré surface of section, which is the standard way physicists detect the presence of chaos in systems (Goldstein et al., 2002). Such diagrams are made by taking a two-dimensional section of the phase-space, simulating the trajectories across large time-steps, and seeing where the trajectories intersect the section (Goldstein et al., 2002). We show an example of this below in Figure 2.1 and Figure 2.2.

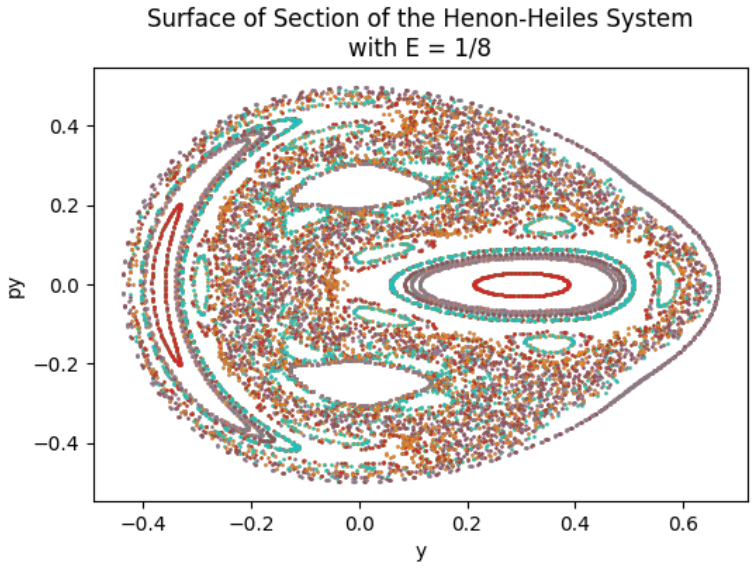

Figure 2.1: Poincaré surface of section for $E = \frac{1}{8}$

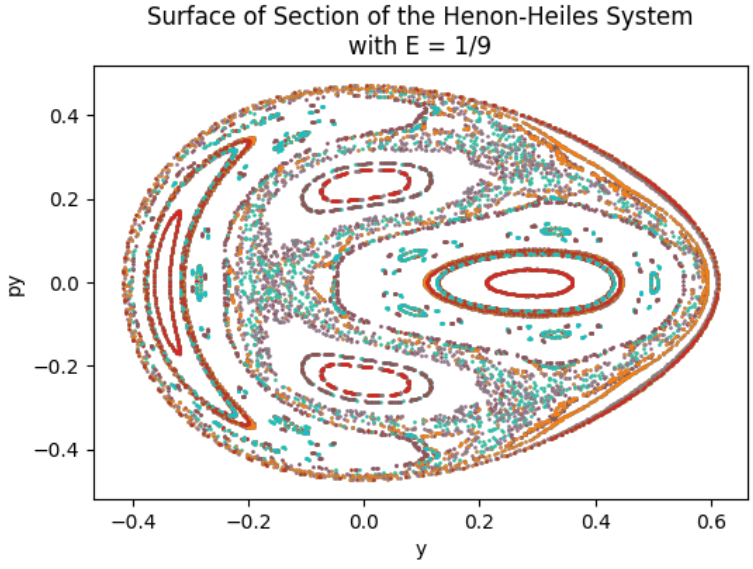

Figure 2.2: Poincaré surface of section for $E = \frac{1}{9}$

In order to construct such diagrams, we would need the use of numerical integrators. These are often in the form of symplectic integrators, due to the variables that the Hamiltonian depend on. Such integration schemes will be discussed in Section 4.2.

2.2.3 Lyapunov Exponents and Local Growth Rates

In this project, however, we will attempt to quantify such degrees of chaos with the local growth rate. This is an alternative to the Lyapunov exponent, which has the following definition of the aforementioned quantity as given in Goldstein et al. (2002):

Definition 2.2.2 (Lyapunov exponent (Goldstein et al., 2002))**.** The Lyapunov exponent can be defined in phase-space as the quantity λ in the equation below:

$$S(t) = S(0)\exp(\lambda t) \tag{2.21}$$

where $S(t)$ denotes the distance between two trajectories in phase-space, t denotes the time taken since the start, and $S(0)$ refers to a small distance between the trajectories at $t = 0$.

We can hence also write the Lyapunov exponent as (Datseris, 2018):

$$\lambda = \frac{1}{t}\log\left(\frac{S(t)}{S(0)}\right) \tag{2.22}$$

However, as we are not interested in merely computing the maximal Lyapunov exponent for a given system, we will need a different, localised quantity. This means that the aforementioned definition must be applied to trajectories that are perturbed and evolved in parallel. Such a quantity is also known as the "local growth rate" or the "nonlinear local Lyapunov exponent" (Datseris, 2018); we shall use the former term to describe the quantity throughout the rest of this report.

2.3 Hamiltonian Neural Networks

Hamiltonian neural networks (HNNs) are a modification of the standard neural network which predicts a Hamiltonian-like value. Below we give the loss function described in Greydanus et al. (2019).

Definition 2.3.1 (Standard HNN loss function)**.** The standard Hamiltonian neural network loss function is denoted as $\mathcal{L}_{HNN}$ and defined as:

$$\mathcal{L}_{HNN} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}} + \frac{\partial \mathbf{p}}{\partial t} \right\|_2 \tag{2.23}$$

This loss function is used during the training of the model, which generates a Hamiltonian-like value as its main output. This output will then be used in conjunction with automatic differentiation to generate $\mathcal{L}_{HNN}$.

HNNs are described to be able to predict a Hamiltonian-like value (Greydanus et al., 2019). This means that they produce values that are of the same scale as the ground truth Hamiltonian but is a constant multiple of said quantity (Greydanus et al., 2019). In comparison to a neural network, which outputs both $\frac{\partial \mathcal{H}}{\partial \mathbf{p}} = \frac{\partial \mathbf{q}}{\partial t}$ and $-\frac{\partial \mathcal{H}}{\partial \mathbf{q}} = \frac{\partial \mathbf{p}}{\partial t}$, they are designed to conserve the Hamiltonian-like value across the trajectory (Greydanus et al., 2019). We shall describe them in greater detail in Chapter 3, where we will also be discussing the limitations of said work.

Chapter 3

Related Work

There have been further research done on networks that have attempted to predict the behaviour of chaotic dynamical systems. We shall categorise the proposed methods that have been used to do the above into whether they use physics-informed models or other types of models.

3.1 Hamiltonian Neural Networks

Hamiltonian neural networks were introduced in Greydanus et al. (2019); they were described as outperforming baselines that had $(\mathbf{q}, \mathbf{p})$ as inputs and $\left(\frac{\partial \mathbf{q}}{\partial t}, \frac{\partial \mathbf{p}}{\partial t}\right)$ as outputs. This was observed across different datasets, including a canonical oscillator, ideal pendulum, as well as a real-world pendulum dataset (Greydanus et al., 2019). It was also observed that the network performed well on a larger two-body problem dataset (Greydanus et al., 2019). Such conclusions were attained by measuring the MSE between coordinates they were evolved from initial states, the total HNN-conserved quantity, as well as the total energy.

Said work in Greydanus et al. (2019), however, was limited by the fact that there were only four datasets in total. Moreover, the datasets chosen were not of particularly high dimensions; the two-body problem, which was the most complex dataset chosen, was described to only have "four (p, q) pairs" (Greydanus et al., 2019). As such, we believe that there ought to be more systems explored in our project.

3.1.1 Separable Hamiltonian Neural Networks

Separable HNNs were introduced in Khoo et al. (2024) alongside modifications to the HNN architecture that introduced observational, learning, and/or inductive biases. In the same paper, HNNs with observational and inductive biases were shown to outperform the baseline (i.e., original) HNN (Khoo et al., 2024) introduced in Greydanus et al. (2019). This conclusion was obtained by analysing the MSE between the ground truth and predicted Hamiltonians, as well as the mean magnitude of error between the ground truth and predicted time derivatives.

We believe that the work in Khoo et al. (2024) can be adapted to our needs. Firstly, we have similar systems to be experimented upon; this allows us to compare the performances we attain from training. Moreover, we believe that the models developed in Khoo et al. (2024) are adapted to our needs, as they have increased performance compared to Greydanus et al. (2019). This is crucial as we would like to focus more on the ability of such networks to detect and measure chaotic behaviour in different regions, which necessitates the model to minimise any errors that result from limitations in its architecture. As such, we describe the various models pertinent to our project below.

Pinned HNN

This type of model uses the basic structure of a HNN with the modified loss function as described in Khoo et al. (2024) and restated below:

Definition 3.1.1 (Pinned HNN loss function (Khoo et al., 2024))**.** The loss function used to train the Hamiltonian neural network, $\mathcal{L}_{HNN}$ is defined as:

$$\mathcal{L}_{HNN} = \sum_{k=0}^{2} c_k l_k = c_0 \left\| \mathcal{H}_\theta(\mathbf{0}) - \mathcal{H}_0 \right\|^2 + c_1 \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \right\|^2 + c_2 \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}} - \frac{\partial \mathbf{p}}{\partial t} \right\|^2 \tag{3.1}$$

where $\mathcal{H}_0$ is a fixed value of the Hamiltonian at some point, usually where $(\mathbf{q}, \mathbf{p}) = (\mathbf{0}, \mathbf{0})$, $\mathcal{H}_\theta$ is the predicted Hamiltonian-like value, and (c_0, c_1, c_2) being a vector of hyperparameter constants, usually set to $(1, 1, 1)$. As we can see, there is the addition of a "pinning term" $\mathcal{H}_0$ that tries to guide the HNN.

HNN with Learning Bias

In order to encode the existence of additive separability, Khoo et al. (2024) included an additional term l_3 as described below:

$$l_3 = \left\| \frac{\partial^2 \mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial \mathbf{q} \partial \mathbf{p}} \right\|^2 \tag{3.2}$$

$$\mathcal{L}_{HNN} = \sum_{k=0}^{3} c_k l_k \tag{3.3}$$

where l_0, l_1, l_2 have their normal definitions as described in Definition 3.1.1.

This additional term will naturally be 0 in the case of additive separable systems. Such HNNs that have this bias will be denoted as HNN-L (Khoo et al., 2024).

HNN with Inductive Bias

Inductive bias, as defined in Khoo et al. (2024), is integrated into the model via the architecture itself. In such HNNs, there are two separate sub-networks each computing $f(\mathbf{q})$ and $g(\mathbf{p})$, which are then combined at the final layer. In essence, we have the forward propagation of the network as:

$$\mathcal{H}_\theta = h(f(\mathbf{q}) \oplus g(\mathbf{p})) \tag{3.4}$$

$$\text{where } h(X) = AX + b \text{ and } \oplus \text{ is the concatenation operator} \tag{3.5}$$

Such HNNs that have this bias will be denoted as HNN-I (Khoo et al., 2024).

HNN with Learning and Inductive Bias

We are able to combine the presence of both learning and inductive biases in our model. Such networks will be denoted as HNN-LI (Khoo et al., 2024).

3.2 Hypothesis

We hypothesise that the quality of the predictions given by the various models will be related to the degree of chaotic behaviour as defined in Subsection 2.2.3. This means that an increase in the local growth rate relates to a decrease in the quality of the predictions. Such a quantity that defines the quality of the prediction shall be defined in greater detail in Chapter 4.

Chapter 4

Methodology

We have used the following steps to approach our research:

1. Identification and literature review of additive separable dynamical systems;
2. Data generation for selected dynamical systems;
3. Selecting, designing, training, and hyperparameter tuning of models; and
4. Analysis and evaluation of model performance.

We shall describe steps 1 to part of 3 in this chapter; the rest shall form part of Chapter 5.

4.1 Additive Separable Dynamical Systems

Identification of the systems was mainly done based on Khoo et al. (2024) and their associated GitHub repository. We describe the systems used in greater detail below.

4.1.1 Pendulum

The pendulum is a classic dynamical system that is easily described and non-chaotic (i.e., it does not have any chaotic regions). We describe the pendulum below.

In Figure 4.1, we have a mass m_1 attached to a string of length l_1. In our case, however, the mass and length of the string are taken as 1. We can hence write the Hamiltonian and related equations of motion as (Khoo et al., 2024):

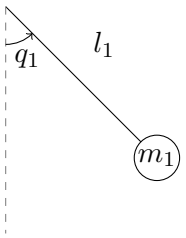

Figure 4.1: Diagram of a pendulum.

$$\mathcal{H}(q_1, p_1) = \frac{1}{2}p_1^2 + (1 - \cos q_1) \tag{4.1}$$

$$\nabla_{\mathbf{p}}\mathcal{H}(q_1, p_1) = p_1 \tag{4.2}$$

$$\nabla_{\mathbf{q}}\mathcal{H}(q_1, p_1) = \sin q_1 \tag{4.3}$$

4.1.2 Trigonometric

This is an arbitrary, non-chaotic system created with the following Hamiltonian and Hamilton's equations (Khoo et al., 2024):

$$\mathcal{H}(q_1, p_1) = \sin^2(q_1) + \cos^2(p_1) \tag{4.4}$$

$$\nabla_{\mathbf{p}}\mathcal{H}(q_1, p_1) = -2\cos(p_1)\sin(p_1) \tag{4.5}$$

$$\nabla_{\mathbf{q}}\mathcal{H}(q_1, p_1) = 2\cos(q_1)\sin(q_1) \tag{4.6}$$

4.1.3 Arctangent

This is also an arbitrary, non-chaotic system created with the following Hamiltonian and Hamilton's equations (Khoo et al., 2024):

$$\mathcal{H}(q_1, p_1) = \arctan(q_1^2) + \arctan(p_1^2) \tag{4.7}$$

$$\nabla_{\mathbf{p}}\mathcal{H}(q_1, p_1) = \frac{2p_1}{p_1^4 + 1} \tag{4.8}$$

$$\nabla_{\mathbf{q}}\mathcal{H}(q_1, p_1) = \frac{2q_1}{q_1^4 + 1} \tag{4.9}$$

4.1.4 Logarithm

This is the last arbitrary, non-chaotic system created with the following Hamiltonian and Hamilton's equations (Khoo et al., 2024):

$$\mathcal{H}(q_1, p_1) = q_1 - \log(q_1^2) - p_1 + \log(p_1^3) \tag{4.10}$$

$$\nabla_{\mathbf{p}}\mathcal{H}(q_1, p_1) = -1 + \frac{3}{p_1} \tag{4.11}$$

$$\nabla_{\mathbf{q}}\mathcal{H}(q_1, p_1) = 1 - \frac{2}{q_1} \tag{4.12}$$

4.1.5 Anisotropic Oscillator

The two-dimensional anisotropic oscillator system is governed by the following equations (Khoo et al., 2024):

$$\mathcal{H}(q_1, q_2, p_1, p_2) = \sqrt{p_1^2 + p_2^2 + 1} + \frac{1}{2}(q_1^2 + q_2^2) + \frac{1}{4}(0 \times q_1^4 + 0.05 \times q_2^4) \tag{4.13}$$

$$\nabla_{\mathbf{p}}\mathcal{H}(q_1, q_2, p_1, p_2) = \left(\frac{p_1}{\sqrt{p_1^2 + p_2^2 + 1}}, \frac{p_2}{\sqrt{p_1^2 + p_2^2 + 1}}\right) \tag{4.14}$$

$$\nabla_{\mathbf{q}}\mathcal{H}(q_1, q_2, p_1, p_2) = \left(q_1, q_2 + 0.05 \times q_2^3\right) \tag{4.15}$$

4.1.6 Hénon-Heiles

The Hénon-Heiles system (Henon & Heiles, 1964) is defined by its Hamiltonian:

$$\mathcal{H}(q_1, q_2, p_1, p_2) = \frac{1}{2}\left(p_1^2 + p_2^2\right) + \frac{1}{2}\left(q_1^2 + q_2^2\right) + \lambda\left(q_1^2 q_2 - \frac{q_2^3}{3}\right) \tag{4.16}$$

where λ is taken to be 1. The system is also governed by its Hamilton's equations:

$$\nabla_{\mathbf{p}}\mathcal{H}(q_1, q_2, p_1, p_2) = (p_1, p_2) \tag{4.17}$$

$$\nabla_{\mathbf{q}}\mathcal{H}(q_1, q_2, p_1, p_2) = \left(q_1 + 2q_1 q_2, q_2 + q_1^2 - q_2^2\right) \tag{4.18}$$

4.1.7 Toda Lattice

The Toda Lattice is a dynamical system that represents a one-dimensional "nonlinear lattice system" (Ford, Stoddard, & Turner, 1973) (Toda, 1967). Its complete Hamiltonian for N particles can be written as (Teschl, 2009):

$$\mathcal{H}(\mathbf{q},\mathbf{p}) = \sum_{i=1}^{n} \left(\frac{1}{2} p_n^2(t) + V(q_{n+1}(t) - q_n(t)) \right) \tag{4.19}$$

In the spirit of simplicity, we shall use a simplified version of the Toda Lattice equations with $n = 3$ as seen in Ford et al. (1973) and Khoo et al. (2024):

$$\begin{aligned} \mathcal{H}(q_1, q_2, q_3, p_1, p_2, p_3) = \frac{1}{2}\left(p_1^2 + p_2^2 + p_3^2\right) + \exp(q_1 - q_2) \\ + \exp(q_2 - q_3) + \exp(q_3 - q_1) - 3 \end{aligned} \tag{4.20}$$

The governing Hamilton's equations are as follows:

$$\nabla_{\mathbf{p}} \mathcal{H}(\mathbf{z}) = (p_1, p_2, p_3) \tag{4.21}$$

$$\begin{aligned} \nabla_{\mathbf{q}} \mathcal{H}(\mathbf{z}) = (\, \exp(q_1 - q_2) - \exp(q_3 - q_1), \exp(q_2 - q_3) - \exp(q_1 - q_2), \\ \exp(q_3 - q_1) - \exp(q_2 - q_3)) \end{aligned} \tag{4.22}$$

4.1.8 Coupled Oscillator

A coupled oscillator is a modification on the original oscillator, a completely integrable system made of a spring attached to a wall and a mass on one side. Hence, the Hamiltonian and other governing equations are also similar to that of the original oscillator, as seen below:

$$\mathcal{H}(\mathbf{q},\mathbf{p}) = \frac{1}{2}\left[\sum_{i=1}^{n} p_i^2 + \sum_{i=2}^{n} (q_i - q_{i-1})^2 \right] \tag{4.23}$$

$$\nabla_{\mathbf{p}} \mathcal{H}(\mathbf{q},\mathbf{p}) = (p_1, p_2, ..., p_n) \tag{4.24}$$

$$\begin{aligned} \nabla_{\mathbf{q}} \mathcal{H}(\mathbf{q},\mathbf{p}) =& (-(q_2 - q_1), (q_2 - q_1) - (q_3 - q_2), ..., \\ & (q_{n-1} - q_{n-2}) - (q_n - q_{n-1}), (q_n - q_{n-1})) \end{aligned} \tag{4.25}$$

We use two versions of this system in our research: one system with $n = 3$ and another with $n = 10$.

4.2 Data-generation for Selected Dynamical Systems

In order to generate the data for each of our selected dynamical systems, we implemented several numerical integration methods, such as the symplectic Euler scheme, the Störmer-Verlet integrator, and the fourth-order symplectic integrator by adapting from Khoo et al. (2024). Common amongst all of these is the nature of the integrators: they are able to conserve a slightly modified Hamiltonian (Wikipedia contributors, 2025b).

The data was generated by using `JAX` (Bradbury et al., 2018), a scientific computing and machine learning library which allows for parallelisation using GPU-based accelerators with CUDA (Nickolls, Buck, Garland, & Skadron, 2008). However, in our experiments, we only used data generated by the Störmer-Verlet integrator, as we are dealing with additive separable systems. Nevertheless, we shall describe all three symplectic integration schemes below.

4.2.1 Symplectic Euler Scheme

The symplectic Euler scheme is a set of symplectic methods of order 1 (Hairer, 2010). We shall use the scheme as given below in our implementation:

$$\begin{aligned} p_{n+1} &= p_n - h\nabla_q \mathcal{H}(q_{n+1}, p_n) \\ q_{n+1} &= q_n + h\nabla_p \mathcal{H}(q_{n+1}, p_n) \end{aligned} \tag{4.26}$$

As the equations are coupled in terms of each other, we will need to use fixed-point iteration to compute the values of q_{n+1} and p_{n+1}. This scheme is able to be used for non-separable Hamiltonian systems (Hairer, 2010).

4.2.2 Störmer-Verlet Integrator

The Störmer-Verlet scheme is a set of symplectic methods of order 2 (Hairer, 2010). We shall use the equations below in our implementation:

$$
\begin{aligned}
q_{n+1/2} &= q_n + \frac{h}{2}\nabla_p \mathcal{H}(q_{n+1/2}, p_n) \\
p_{n+1} &= p_n - \frac{h}{2}\left(\nabla_q \mathcal{H}(q_{n+1/2}, p_n) + \nabla_q \mathcal{H}(q_{n+1/2}, p_{n+1})\right) \\
q_{n+1} &= q_{n+1/2} + \frac{h}{2}\nabla_p \mathcal{H}(q_{n+1/2}, p_{n+1})
\end{aligned} \tag{4.27}
$$

As in the symplectic Euler scheme, we will need to use fixed-point iteration. This scheme, however, is only able to be used for separable Hamiltonian systems (Hairer, 2010).

4.2.3 Fourth-order Symplectic Integrator

The fourth-order symplectic integrator (Forest & Ruth, 1989) can be formulated as follows:

$$\mathbf{p}_{i+1} = \mathbf{p}_i - c_i t V(\mathbf{q}_i) \tag{4.28}$$

$$\mathbf{q}_{i+1} = \mathbf{q}_i + d_i t T(\mathbf{p}_{i+1}) \tag{4.29}$$

$$\tag{4.30}$$

for $i = 1, ..., 4$, where $(\mathbf{q}_0, \mathbf{p}_0)$ are the initial conditions and $(\mathbf{q}_4, \mathbf{p}_4)$ is the result after a time step of t (Forest & Ruth, 1989). The coefficients can then be computed as (Forest & Ruth, 1989):

$$
\begin{cases}
c_1 = x + \frac{1}{2}, & d_1 = 2x + 1 \\
c_2 = -x, & d_2 = -4x - 1 \\
c_3 = -x, & d_3 = 2x + 1 \\
c_4 = x + \frac{1}{2}, & d_4 = 0
\end{cases} \tag{4.31}
$$

$$
\text{where } x = \frac{2^{1/3} + 2^{-1/3} - 1}{6} \tag{4.32}
$$

In the generation of the data used in our experiments, we first generated n points p of dimension d according to the system. Each dimension of every point is generated from a random uniform distribution according to the range of values previously defined for each dimension.

Moreover, deviated initial points $p' = p + \varepsilon r$, $r \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ were generated. Here, ε was manually chosen as an argument. Both of these were evolved for N time-steps using the Störmer-Verlet integrator to produce trajectories $t \in \mathbb{R}^{N \times d}$ and $t' \in \mathbb{R}^{N \times d}$. Thereafter, the last points in the evolved trajectories were used in the computation of the local growth rate λ.

Three datasets that each represent the training, validation, and test sets were generated for every system. We used $n = 1500, N = 512, \varepsilon \in \{10^{-6}, 10^{-4}\}$. A split of 80-10-10 was used for the datasets, respectively. Note that this splitting was done on the n points, i.e., n points were generated according to the above paragraph before being split.

4.3 Selecting and designing models

In this project, we have decided to use five different models: a multilayer perceptron (MLP), standard HNN, HNN with learning bias, HNN with inductive bias, as well as HNN with both learning and inductive bias. The latter three models are derived from (Khoo et al., 2024). All designs of the models will be done using `PyTorch` (Paszke et al., 2019), which allows for GPU-based accelerators to be used. As we have previously described the HNN variants in Subsection 3.1.1, we shall only restate them in short form together with any fixed hyperparameters below.

Multilayer Perceptron

A standard neural network that takes in the phase-space coordinates and regresses on the Hamiltonian was used as a baseline. This model was trained with standard backpropagation, without the additional information provided by the derivatives of the Hamiltonian with respect to the inputs. Such backpropagation will use the MSE loss on the predicted and ground truth Hamiltonian as the loss function as defined below:

$$MSE(\hat{\mathcal{H}}, \mathcal{H}) = \frac{1}{B} \sum_{i=1}^{B} \left(\hat{\mathcal{H}}^{(i)} - \mathcal{H}^{(i)} \right)^2 \tag{4.33}$$

where B is the batch size of the set that the model is currently being trained on.

This model will be denoted as `mlp`.

Pinned HNN

As described before, we shall use the Pinned HHN (Khoo et al., 2024) with $(c_0, c_1, c_2) = (1, 1, 1)$ as the second model in our comparisons. This model type will be denoted as `hnn`.

HNN with Learning Bias

The HNN with learning bias (HNN-L) (Khoo et al., 2024) shall be used as the third model in our experiments. We shall fix (c_0, c_1, c_2) as previously described, but allow c_3 to become another hyperparameter. This model will be denoted as `hnn-l`.

HNN with Inductive Bias

The HNN with inductive bias (HNN-I) (Khoo et al., 2024) will be the fourth model used for this project. Again, (c_0, c_1, c_2) will be fixed to the aforementioned values. Moreover, we shall only use this type of model with a learnable final linear layer and bias. In our experiments, we have divided the number of neurons used in each hidden dimension for the sub-networks by 2. This model will be denoted as `hnn-i`.

HNN with Learning and Inductive Bias

HNNs that combine both learnable and inductive biases (Khoo et al., 2024) will be used as the fifth model for this project. (c_0, c_1, c_2) will be fixed to the aforementioned values, while c_3 will be a hyperparameter as previously described. Other aforementioned specifications that apply to `hnn-i` shall apply to this model as well. This model will be denoted as `hnn-li`.

Chapter 5

Experiments

In this chapter, we will describe the details of our experiments and provide a discussion on its results.

5.1 Training and Hyperparameter Tuning

Experiments were ran on each of the systems and model types described earlier. For each pair of these, we conducted a random hyperparameter search on several hyperparameters, including the size of each hidden dimension, number of hidden layers, dropout probability, initialization of network layers, activation function for each layer (except the output), learning rate, and strength of c_3 in the loss function.

Each system, model pair had hyperparameter tuning done with 64 random hyperparameter samples, each of them selected according to Table A.1. Such training and hyperparameter tuning were done using the `ASHAScheduler` class provided by `Ray Tune`, which implements the Asynchronous Successive Halving Algorithm (ASHA) (Li et al., 2020). Each hyperparameter sample was trained for a minimum of 4 epochs up to a maximum of 8 epochs, with the reduction factor set to 2. The best hyperparameter was then selected with the use of the MSE metric described below:

$$\text{MSE}(\hat{\mathcal{H}}, \mathcal{H}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{B}\sum_{j=1}^{B}\left(\hat{\mathcal{H}}^{(j)} - \mathcal{H}^{(j)}\right)^2 \tag{5.1}$$

where $\hat{\mathcal{H}}$ and $\mathcal{H}$ are the predicted and ground truth Hamiltonians in the validation set, respectively. Here, we take m to be the number of batches used in validation and B to be the batch size. This is in line with one of the performance metrics used in Khoo et al. (2024).

After selection of the optimal hyperparameter sample that was produced, we trained such sample on the combined training and validation set for at least another 4 epochs up to a maximum of 8 epochs. The best checkpoint's weights, alongside the best hyperparameter sample, were then saved for further analysis. All aforementioned training and hyperparameter tuning were done on the SoC Compute Cluster using NVIDIA A100 and H100 GPUs that are able to run multiple experiments simultaneously.

5.2 Analysis and Evaluation of Model Performance

We will be analysing our models with the use of several metrics. Analysis of the selected hyperparameters for each model and pair will be done with the use of graphs showing various quantities against the local growth rate. The quantities used include the moving-average mean E_H and E_V, whose definitions are similar to that found in Khoo et al. (2024). We define them below.

$$E_H = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{\mathcal{H}}^{(i)} - \mathcal{H}^{(i)}\right)^2 \tag{5.2}$$

$$E_V = \frac{1}{m}\sum_{i=1}^{m}E_V^{(i)} \tag{5.3}$$

$$\text{where } E_V^{(j)} = \frac{\sqrt{\|\hat{\dot{q}}^{(j)} - \dot{q}^{(j)}\|_2^2 + \|\hat{\dot{p}}^{(j)} - \dot{p}^{(j)}\|_2^2}}{\sqrt{\|\dot{q}^{(j)}\|_2^2 + \|\dot{p}^{(j)}\|_2^2}} \tag{5.4}$$

Each of these metrics are evaluated for each local growth rate computed. That is, m is equal to the number of datapoints for each local growth rate.

In order to better analyse the predictive power of our models, we decided on using an additional metric known as the coefficient of determination (Wikipedia contributors, 2025a), defined below:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{5.5}$$

$$\text{where } SS_{res} = \sum_{i=1}^{m} \left(\hat{\mathcal{H}}^{(i)} - \mathcal{H}^{(i)}\right)^2 \tag{5.6}$$

$$\text{and } SS_{tot} = \sum_{i=1}^{m} \left(\mathcal{H}^{(i)} - \bar{\mathcal{H}}\right)^2 \tag{5.7}$$

The R^2 metric has a maximum value of 1 and can be negative, if the model performs arbitrarily poorly.

Note that the predicted Hamiltonians $\hat{\mathcal{H}}^{(i)}$ used above are shifted from the original predictions. That is,

$$\hat{\mathcal{H}}^{(i)}_{\text{shifted}} = \hat{\mathcal{H}}^{(i)}_{\text{original}} + (\bar{\mathcal{H}}^{(i)} - \bar{\mathcal{H}}^{(i)}_{\text{original}}) \tag{5.8}$$

This is done in line with the statement found in Greydanus et al. (2019) and reiterated in Chapter 2; that is, such networks predict on the same scale as, and are a constant multiple of, the ground truth Hamiltonian.

Graphs pertinent to the discussion below have been included in Appendix A. Note that the graphs of the `logarithm` system are not present as the models were unable to converge for this system, most likely due to numerical instability.

5.3 Discussion

5.3.1 E_V Against Local Growth Rate Discussion

Across both graphs pictured in Section A.2, we can see that in general, the rolling mean of E_V values increases as the local growth rate increases. This phenomenon is more pronounced for $\varepsilon = 10^{-4}$ than $\varepsilon = 10^{-6}$, which we believe to be due to the nature of the dataset: a larger

ε leads to a greater initial separation, which means that it is more likely for the points to be separated by a larger phase-space distance at the end of their evolution. Both these graphs provide evidence that more chaotic regions tend to cause the quality of the models' predictions to deteriorate.

In general, the various HNN variants tend to have performed better and more consistently for $\varepsilon = 10^{-6}$ than for $\varepsilon = 10^{-4}$. This is an interesting behaviour, as one might expect that a greater range of values available for training would lead to better performance.

5.3.2 E_H Against Local Growth Rate Discussion

Across both graphs pictured in Section A.3, there appears to be increased jitteriness as there is an increase in the local growth rate. This is partially due to the data generation, which was not done using a uniform distribution; however, we believe that this is also due to greater uncertainty in the model in predicting $\hat{\mathcal{H}}$. Hence, both these graphs show that there are lower-quality predictions being made as there is an increase in the local growth rate.

Moreover, it appears that the baseline is tied against the HNN variants in predicting the Hamiltonian. This is to be expected, as the baseline directly predicts the Hamiltonian instead of trying to learn it via a related loss function.

5.3.3 R^2 Against Time-step Discussion

The two graphs pictured in Section A.4 demonstrate that the models are indeed able to learn the Hamiltonian function well across regions. Note that each subgraph is somewhat mirrored vertically; this is expected, as the first half of each subgraph contains the time-evolution of original points, whereas the second half demonstrates the same for the deviated points.

Specifically, we notice that the baseline MLP consistently performs very well. This is again expected, as the MLP was directly trained on predicting the Hamiltonian. Interestingly, some of the variants demonstrated that their performance degraded across the time-steps. We believe that this is due to the objective of the HNN variants in conserving the "symplectic gradient" (Greydanus et al., 2019) instead of the Hamiltonian. We also conjecture that if the models are

trained with more epochs and more advanced loss functions, they will be able to conserve the Hamiltonian much more effectively.

5.3.4 Mean MSE Against Time-step Discussion

The two graphs pictured in Section A.5 provide even stronger evidence that the models have been trained well. It is observed that in each dataset, the models generate similar lines, which implies that they perform similarly well. Indeed, if we compare the various charts against Figure 7 of Khoo et al. (2024), we are able to see that the models here perform competitively in almost all datasets.

We observe that the performance of the models across the various datasets do not differ very much for $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-6}$. It seems to be the case, however, that the latter performs slightly better than the former, supporting our observation in Subsection 5.3.2.

5.3.5 General Discussion

Generally, we find that the models have been trained well. Some architectures tend to perform better with some datasets, but this is expected with such different datasets.

Across the different results, we find that the performance metrics correlate quite well. That is, a worse or jittery E_H often correlates with a worse E_V. This provides evidence that we are able to use either metric in determining the degree of chaotic behaviour present in a region.

We also observe that the R^2 metric from Section A.4 is able to give us a good picture into the trend found in Section A.2. Models that have better R^2 metric values over the time-steps generally perform better and more consistently in terms of their E_V metric. This is expected: the performance of the models over time, where the initial and deviated points diverge, should give an indication of how well the models perform in more chaotic regions.

Chapter 6

Conclusion

We propose that the quality of model predictions relates to the degree of chaos in different systems. This hypothesis is shown to be correct by measuring the value of metrics such as E_H and E_V across the various local growth rates found in a system. That is, degrading model prediction quality relative to the magnitude of the local growth rate inside various systems is observed. Moreover, we are able to see that the predictive quality of the models across time-steps generally correlates with their performance in chaotic regions of different systems.

We further propose several possible avenues for future work. Firstly, it would be good to explore this phenomenon for greater values of ε. This is important to establish that such an observation is general and not by virtue of the dataset itself. Secondly, one should try to use loss functions that pin the entire range of $\mathcal{H}$ values, instead of only doing so for $\mathcal{H}_0$. We believe that this would allow for even better performance on the metrics associated with the measurement of $\mathcal{H}$. Finally, we believe that it is prudent to include the use of additional, more general measures of chaotic behaviour such as the Lyapunov spectra (Datseris, 2018).

References

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., ... Zhang, Q. (2018). *JAX: composable transformations of Python+NumPy programs.* Retrieved from `http://github.com/jax-ml/jax`

Buizza, R. (2002). *Chaos and weather prediction.* `https://www.ecmwf.int/en/elibrary/79859-chaos-and-weather-prediction`. (Accessed: 2025-03-31)

Cline, D. (2021). *Variational principles in classical mechanics.* University of Rochester River Campus Libraries.

Datseris, G. (2018, mar). Dynamicalsystems.jl: A julia software library for chaos and nonlinear dynamics. *Journal of Open Source Software*, *3*(23), 598. Retrieved from `https://doi.org/10.21105/joss.00598` doi: 10.21105/joss.00598

Ford, J., Stoddard, S. D., & Turner, J. S. (1973, 11). On the integrability of the toda lattice. *Progress of Theoretical Physics*, *50*(5), 1547-1560. Retrieved from `https://doi.org/10.1143/PTP.50.1547` doi: 10.1143/PTP.50.1547

Forest, E., & Ruth, R. D. (1989). Fourth-order symplectic integration. *Physica D*, *43*.

Goldstein, H., Poole, C., & Safko, J. (2002). *Classical mechanics.* Addison Wesley.

Greydanus, S., Dzamba, M., & Yosinski, J. (2019). *Hamiltonian neural networks.* Retrieved from `https://arxiv.org/abs/1906.01563`

Hairer, E. (2010). *Lecture 2: Symplectic integrators.* `https://www.unige.ch/~hairer/poly_geoint/week2.pdf`. TU München. (Accessed: 2024-10-28)

Hasselblatt, B., & Katok, A. (2003). *A first course in dynamics: with a panorama of recent developments.* Cambridge University Press.

Henon, M., & Heiles, C. (1964, February). The applicability of the third integral of motion: Some numerical experiments. , *69*, 73. doi: 10.1086/109234

Jumper, J. M., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., ... Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, *596*, 583 - 589. Retrieved from https://api.semanticscholar.org/CorpusID:235959867

Khoo, Z.-Y., Wu, D., Low, J. S. C., & Bressan, S. (2024). *Separable hamiltonian neural networks.* Retrieved from https://arxiv.org/abs/2309.01069

Lemarié, G. (2022). *Introduction to variational principles in physics.* National University of Singpaore. (Accessed: 2024-10-15)

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., & Talwalkar, A. (2020). *A system for massively parallel hyperparameter tuning.* Retrieved from https://arxiv.org/abs/1810.05934

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with cuda. In *Acm siggraph 2008 classes.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1401132.1401152 doi: 10.1145/1401132.1401152

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., ... Zoph, B. (2024). *Gpt-4 technical report.* Retrieved from https://arxiv.org/abs/2303.08774

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). *Pytorch: An imperative style, high-performance deep learning library.* Retrieved from https://arxiv.org/abs/1912.01703

Raissi, M., & Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, *357*, 125-141. Retrieved from https://www.sciencedirect.com/science/article/pii/S0021999117309014 doi: https://doi.org/10.1016/j.jcp.2017.11.039

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

https://www.sciencedirect.com/science/article/pii/S0021999118307125 doi: https://doi.org/10.1016/j.jcp.2018.10.045

Teschl, G. (2009). *The toda lattice.* https://www.mat.univie.ac.at/~gerald/ftp/book-jac/toda.html. (Accessed: 2025-04-01)

Toda, M. (1967). Vibration of a chain with nonlinear interaction. *Journal of the Physical Society of Japan*, *22*(2), 431-436. Retrieved from https://doi.org/10.1143/JPSJ.22.431 doi: 10.1143/JPSJ.22.431

Weisstein, E. W. (2025). *Tautochrone problem.* https://mathworld.wolfram.com/TautochroneProblem.html. (Accessed: 2025-04-01)

Wikipedia contributors. (2025a). *Coefficient of determination — Wikipedia, the free encyclopedia.* https://en.wikipedia.org/w/index.php?title=Coefficient_of_determination&oldid=1277871013. ([Online; accessed 2-April-2025])

Wikipedia contributors. (2025b). *Symplectic integrator — Wikipedia, the free encyclopedia.* https://en.wikipedia.org/w/index.php?title=Symplectic_integrator&oldid=1281895276. ([Online; accessed 2-April-2025])

Zang, X., Iqbal, S., Zhu, Y., Liu, X., & Zhao, J. (2016). Applications of chaotic dynamics in robotics. *International Journal of Advanced Robotic Systems*, *13*(2), 60. Retrieved from https://doi.org/10.5772/62796 doi: 10.5772/62796

Appendix A

Experimental Details

This appendix shall include details on hyperparameter tuning and experimental results.

A.1 Hyperparameters

Hyperparameter	Range
Hidden dimension size	$2^i, i \in \{3, 4, .., 10\}$
Number of hidden layers	$i \in \{1, 2, ..., 7\}$
Dropout probability	$\frac{i}{20}, i \in \{0, ..., 10\}$
Initialization	`kaiming_uniform`, `kaiming_normal`, `xavier_uniform`, `xavier_normal`, `orthogonal`
Activation function	`ReLU`, `Leaky-ReLU`, `SiLU`, `GELU`
Learning rate	$\log(\texttt{LR}) \sim \text{Unif}(10^{-3}, 10^{-2})$
c_3	$\{0.25, 0.5, 0.75, 1.0\}$

Table A.1: Hyperparameter Ranges

A.2 E_V Against Local Growth Rate Results

Figure A.1: Mean moving-average E_V against local growth rate, $\varepsilon = 10^{-4}$

Figure A.2: Mean moving-average E_V against local growth rate, $\varepsilon = 10^{-6}$

A.3 E_H Against Local Growth Rate Results

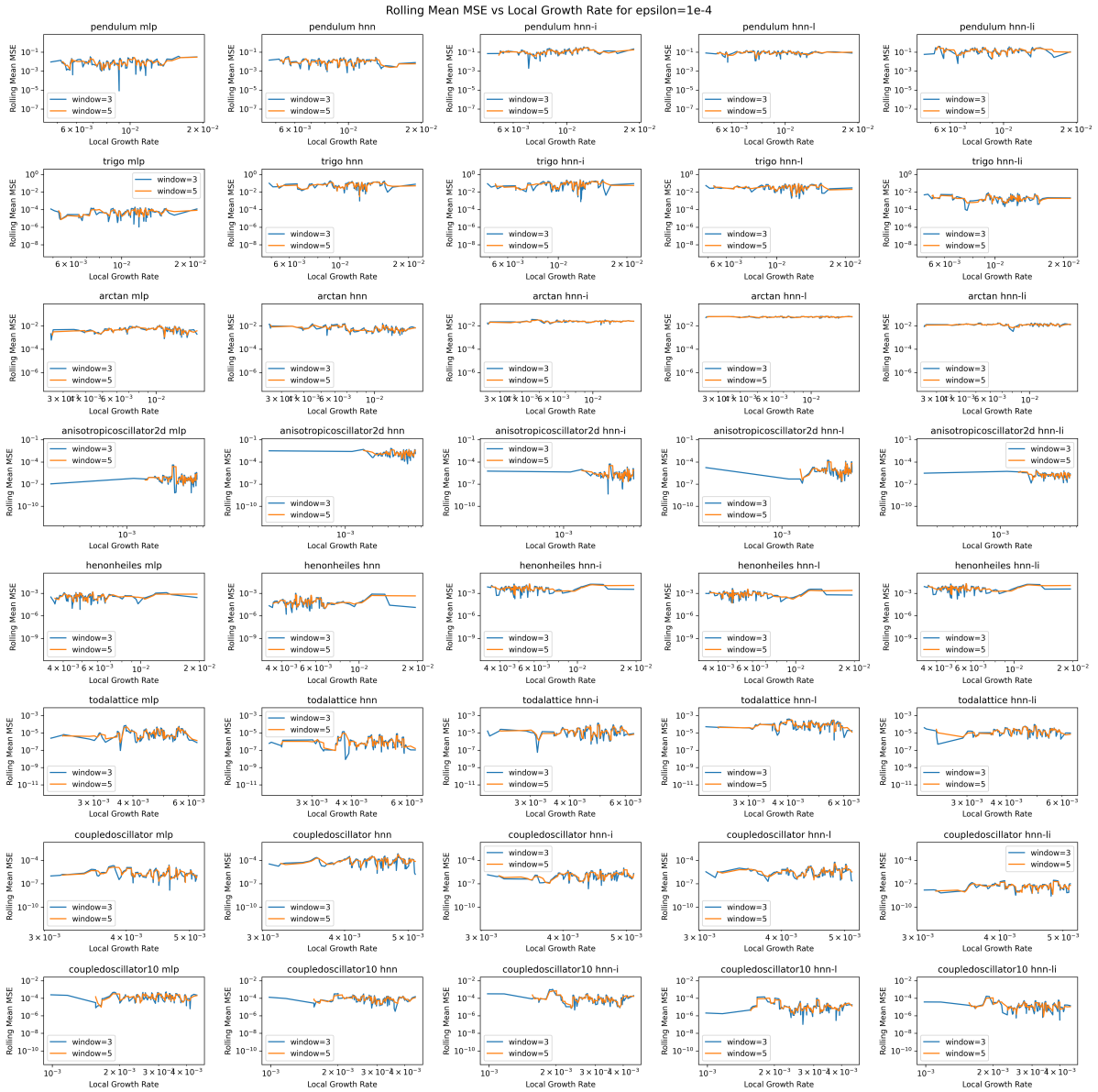

Figure A.3: Mean moving-average MSE against local growth rate, $\varepsilon = 10^{-4}$

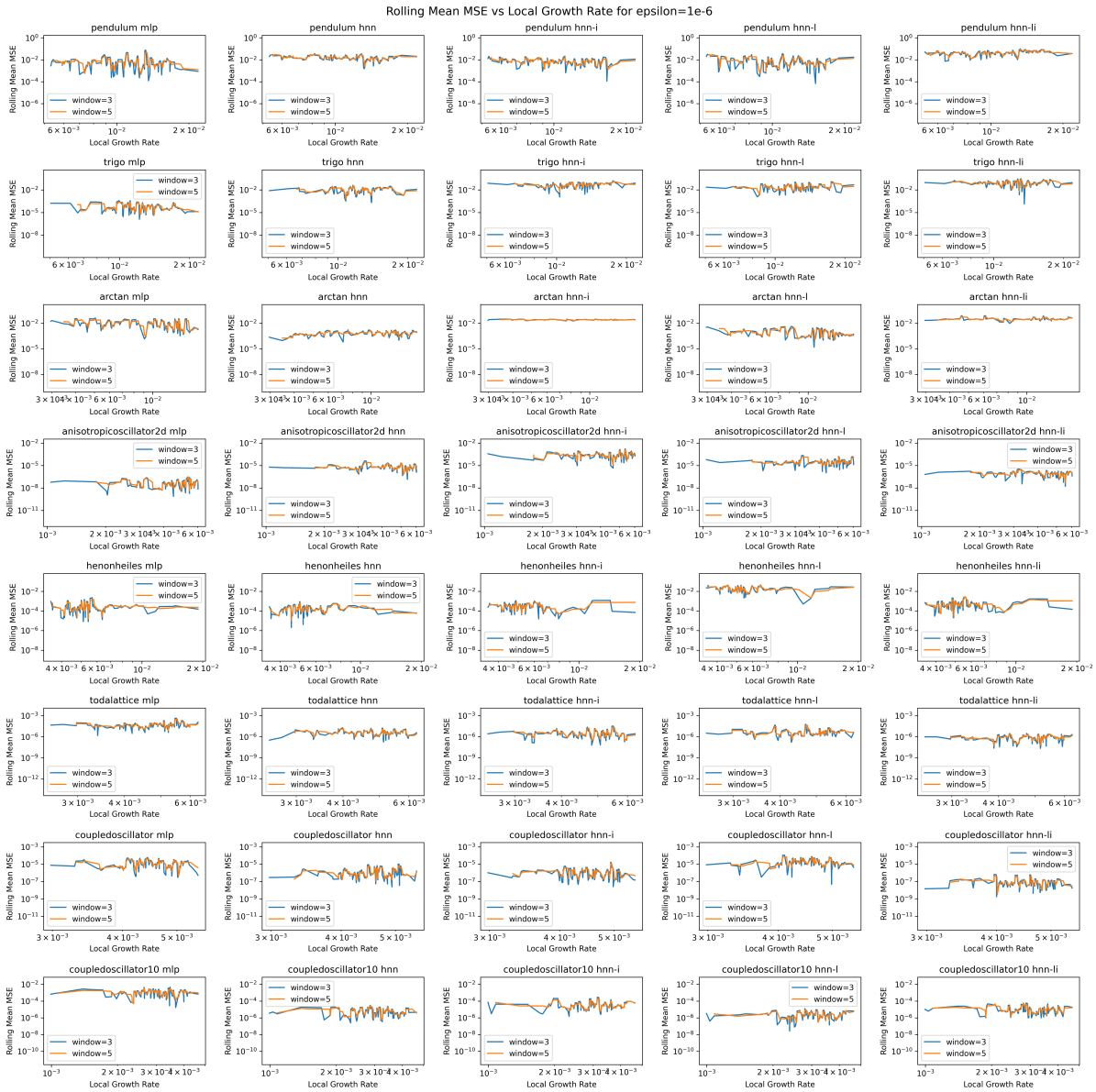

Figure A.4: Mean moving-average MSE against local growth rate, $\varepsilon = 10^{-6}$

A.4 R^2 Against Time-step Results

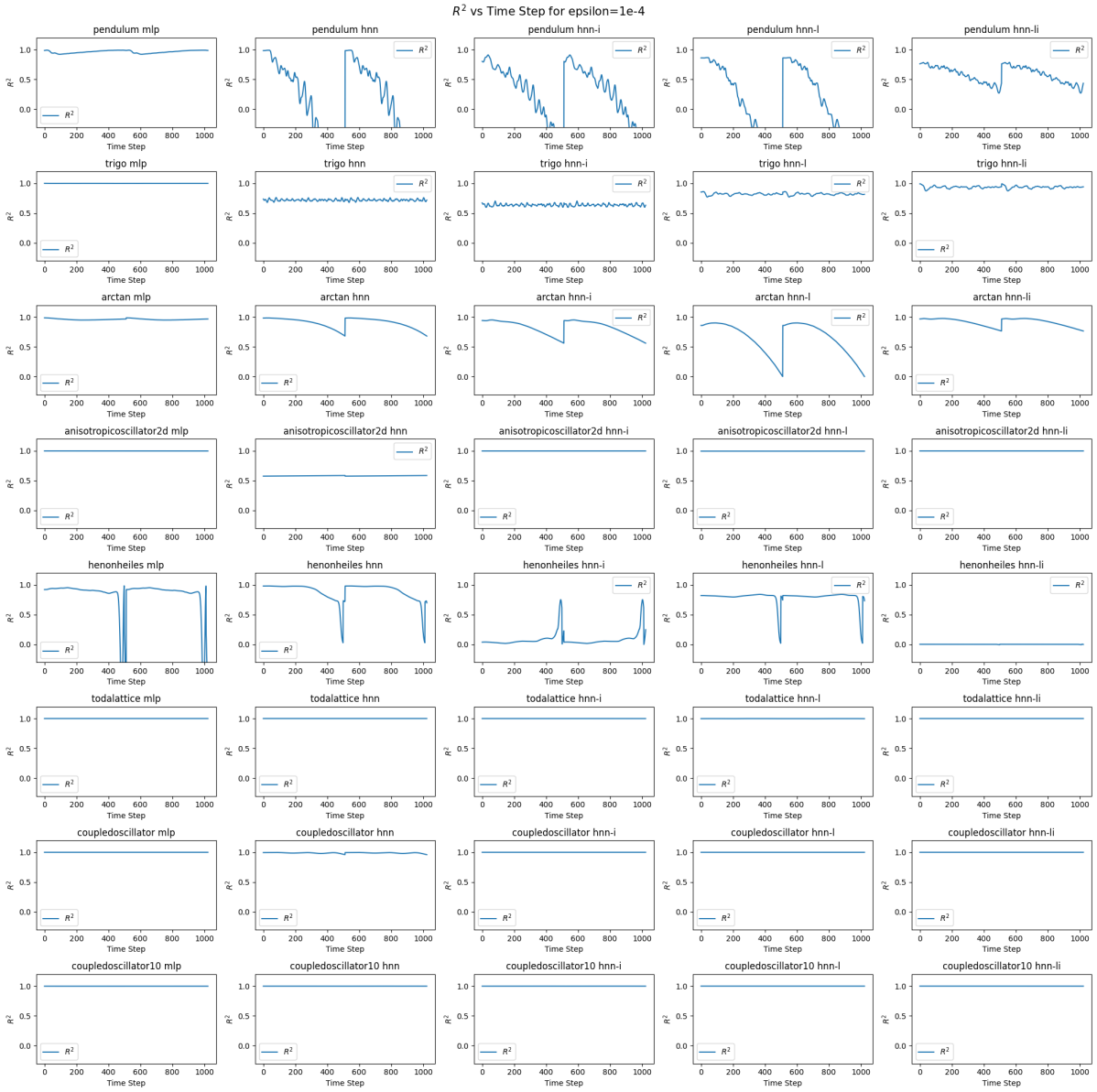

Figure A.5: R^2 vs time-step, $\varepsilon = 10^{-4}$

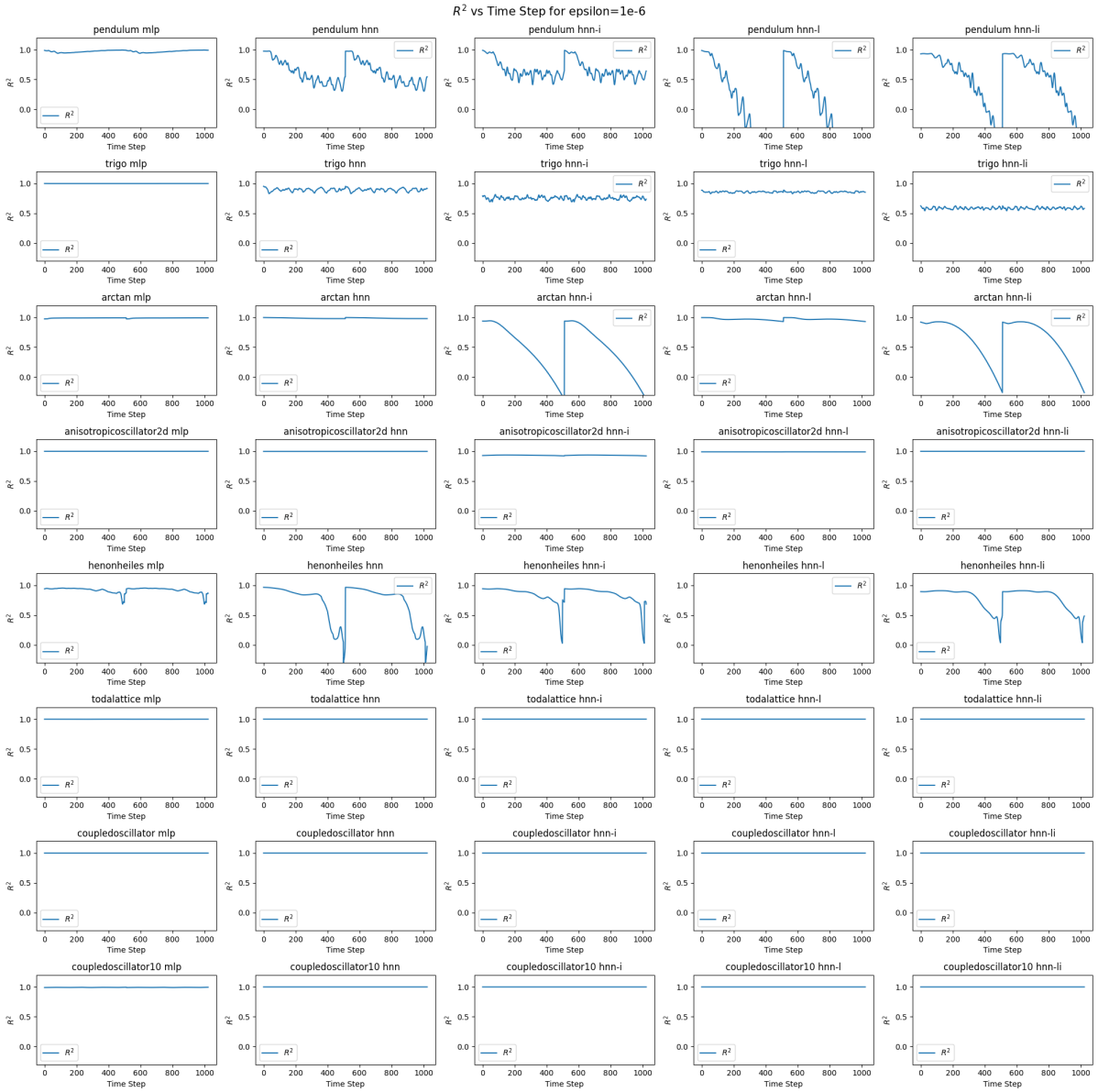

Figure A.6: R^2 vs time-step, $\varepsilon = 10^{-6}$

A.5 Mean MSE Against Time-step Results

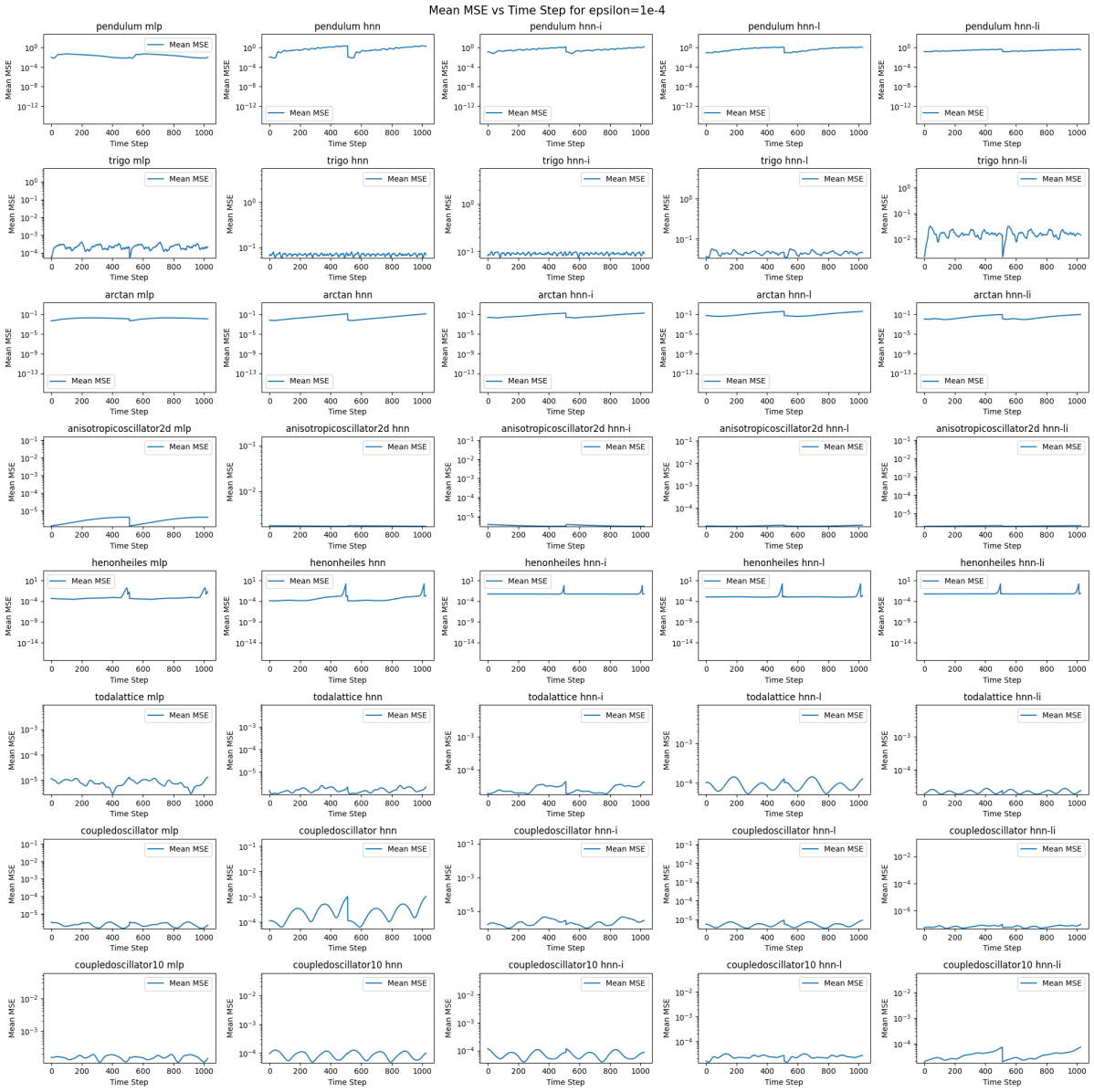

Figure A.7: Mean MSE vs time-step, $\varepsilon = 10^{-4}$

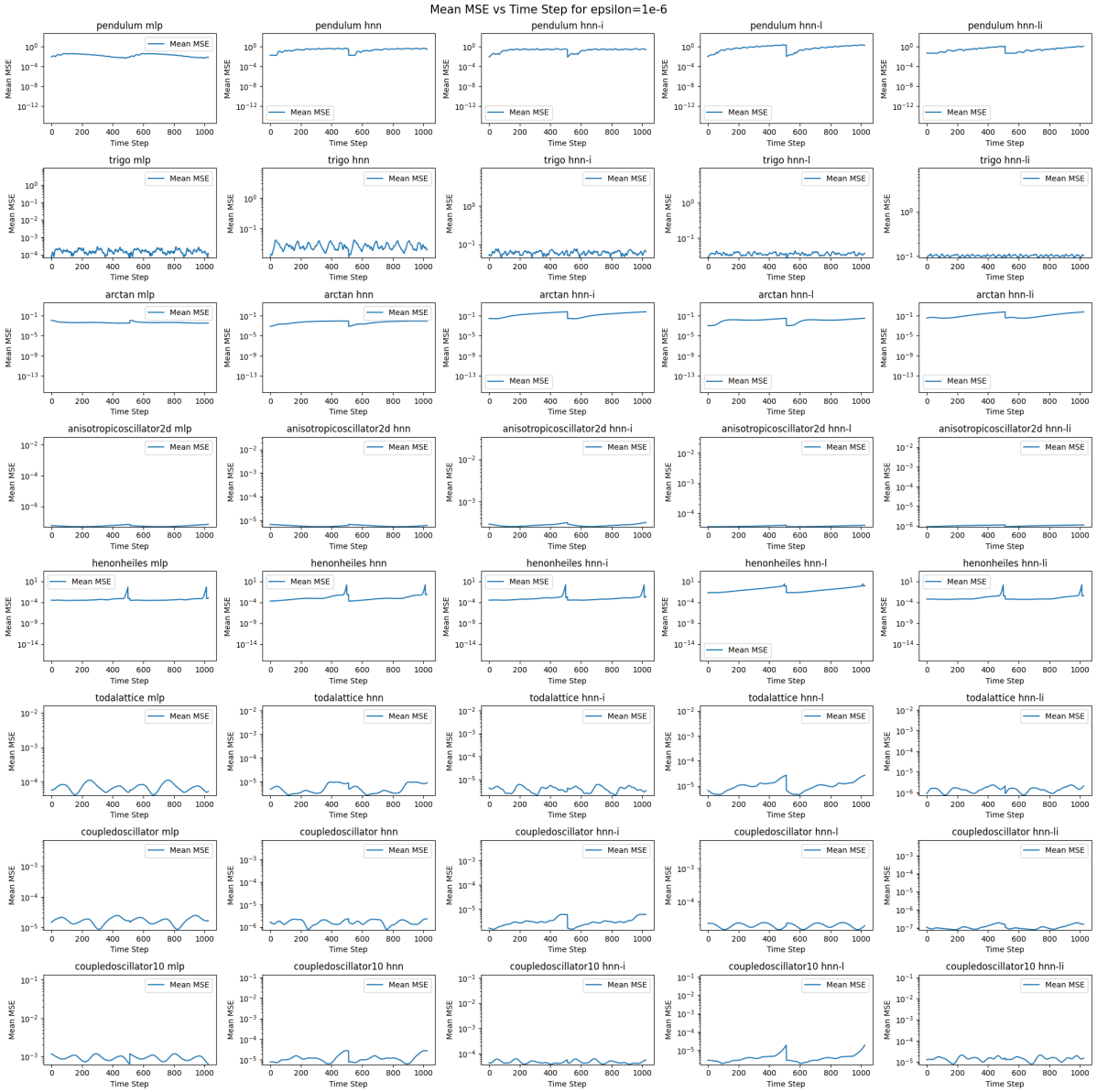

Figure A.8: Mean MSE vs time-step, $\varepsilon = 10^{-6}$