DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Random Feature Hamiltonian Networks for N-Body Problems

Sofia Koufogazou Loukianova

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Random Feature Hamiltonian Networks for N-Body Problems

Randomisierte Neuronale Netze zur Approximation von N-Körper-Problemen

Author:	Sofia Koufogazou Loukianova
Supervisor:	Prof. Dr. Felix Dietrich
Advisor:	M.Sc. Atamert Rahma
Submission Date:	February 2, 2026

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, February 2, 2026 Sofia Koufogazou Loukianova

Acknowledgments

I am deeply grateful to my advisor, Atamert, for his invaluable guidance, encouragement, and continuous support throughout this project. Our productive discussions were essential to shaping this thesis, and his thoughtful feedback and expertise consistently pushed me to refine both the ideas and the results.

Abstract

The gravitational N-body problem is one of the oldest and most fundamental problems in celestial mechanics. In recent years, machine learning approaches have been proposed for modeling such systems, including neural networks trained to approximate the vector field, as well as physics-informed models that incorporate physical priors. Hamiltonian Neural Networks (HNNs) explicitly encode the Hamiltonian structure by learning a scalar Hamiltonian function from data and recovering equations of motion through Hamilton's equations, leading to improved energy conservation and long-term stability compared to unconstrained neural models. Random Feature Hamiltonian Networks accelerate this approach by fixing hidden-layer parameters using data-agnostic or data-driven sampling, thereby reducing computational cost while retaining structure-perserving properties. These methods have shown promising performance on low-dimensional systems such as the Hénon–Heiles system, Lotka–Volterra dynamics, and single and double pendulums. In this work, we generate trajectories for the gravitational two- and three-body problems and train multilayer perceptrons (MLPs), HNNs, and their random-feature counterparts to learn the vector field or the underlying Hamiltonian accordingly. The learned models are evaluated by forward simulation using symplectic integrators. Performance is assessed through long-term trajectory prediction and energy conservation, demonstrating the importance of incorporating Hamiltonian structure when learning dynamical models.

Contents

1 Introduction

Understanding the dynamics of complex physical systems, from molecular interactions to gravitational many-body systems, remains a central challenge in physics. In recent years, neural networks have emerged as a powerful framework for modeling such systems directly from data, with successful applications across a wide range of physical domains [1]. Many real-world systems are governed by unknown or partially known equations of motion, which motivates the development of data-driven approaches. While standard multilayer perceptrons directly learn the system's vector field, more recent machine learning models explicitly incorporate underlying physical structures into the learning process. These physics-informed approaches aim to enforce conservation laws and geometric properties, such as energy conservation and symplectic structure [2, 3, 4, 5, 1, 6]. A natural way to evaluate such models is to apply them to systems with known governing equations and compare their forward-time predictions with those obtained from numerical integration of the true dynamics. The gravitational N-body problem provides a particularly suitable benchmark. The N-body problem is one of the oldest and most fundamental problems in physics and describes the motion of N interacting point masses, such as planets or stars, under Newtonian gravity. While the two-body case admits a closed-form solution, the three-body problem has no general analytic solution; it exhibits chaotic behavior, and is extremely sensitive to initial conditions. Therefore, it provides a challenging test case for evaluating data-driven, physics-informed models.

Hamiltonian Neural Networks (HNNs) represent a prominent approach, as they learn the system's Hamiltonian function rather than directly approximating the vector field, enabling the learned dynamics to respect the symplectic structure of Hamiltonian systems [2, 6]. Greydanus et. al. applied HNNs to gravitational two- and three-body systems, achieving good performance in the two-body case. However, their performance significantly degrades in the chaotic three-body regime, particularly for long-term trajectory prediction [2]. Moreover, the authors used integrators that do not preserve the symplectic structure of the gravitational system, further limiting their long-term accuracy [7]. More recently, Random Feature Hamiltonian Networks (RF-HNNs) have been proposed to accelerate the HNN approach by fixing hidden-layer parameters using data-agnostic or data-driven sampling strategies instead of backpropagation. RF-HNNs have primarily been investigated for systems such as the Hénon–Heiles system, the double pendulum, and spring–mass models, which involve fewer degrees of freedom [3, 5]. This motivates bridging the gap by systematically evaluating RF-HNNs on gravitational two- and three-body systems. In addition to RF-HNNs, we compare three other model classes: a standard multilayer perceptron (MLP), an HNN, and a random-feature MLP (RF-MLP). For random-feature models, we further investigate multiple sampling strategies, including ELM, SWIM. A-SWIM, and U-SWIM. All models are trained

and evaluated using symplectic integration schemes. We construct custom orbit datasets with near-circular two-body trajectories and perturbed periodic three-body orbits based on the Figure-eight and Broucke A1 stable solutions [8, 9]. Our evaluation includes relative test error comparisons and systematic scaling studies with respect to the network width and data set size. Forward simulations are performed from unseen initial conditions, and both trajectory error growth and long-term energy conservation relative to ground-truth orbits are assessed.

This thesis is structured as follows: Section 2 presents the state of the art by introducing the relevant theoretical background on Hamiltonian mechanics and the N-body problem (2.1), followed by a discussion of numerical integration schemes used for simulation (2.2). The section concludes with the mathematical foundations of data-driven models, including classical feed-forward neural networks (2.3), Hamiltonian Neural Networks (2.4), and Random Feature Hamiltonian Networks (2.5). Section 3 describes the training setup for approximating Hamiltonian functions of N-body systems using RF-HNNs (3.1) and details the orbit data generation and evaluation procedures for the two-body (3.2) and three-body (3.3) experiments. Finally, Section 4 concludes with a summary and discussion of the results and outlines directions for future work.

2 State of the art

This chapter reviews the theoretical foundations relevant to learning Hamiltonian dynamics from data. In section 2.1, we introduce the foundations of Hamiltonian dynamics (2.1.1), with a focus on gravitational N-body systems and their geometric and conservation properties (2.1.2). The two-body problem (2.1.3) is presented as a canonical baseline and contrasted with the chaotic three-body problem (2.1.4). In section 2.2, we then review numerical integration techniques for Hamiltonian systems, highlighting the limitations of general-purpose methods (2.2.1) and the advantages of symplectic integrators (2.2.2) for long-term stability. Finally, section 2.3 reviews neural network models for dynamical systems, section 2.4 Hamiltonian Neural Networks, and section 2.5 random feature–based Hamiltonian approaches.

2.1 Hamiltonian Dynamics of N-Body Systems

When simulating physical systems such as the gravitational N-body problem, where total energy is conserved, it is natural to describe the dynamics within a Hamiltonian framework [10]. This formulation explicitly encodes key physical invariants, including total energy, linear and angular momentum, as well as the underlying symplectic structure of phase space. Preserving these invariants is essential for accurately capturing long-term behavior and motivates the use of Hamiltonian-based modeling approaches [7].

2.1.1 The Hamiltonian Formulation of Dynamics

A Hamiltonian system [10] of n degrees of freedom is defined as a system of $2n$ first-order ordinary differential equations of the form

$$\dot{\mathbf{z}} = J\nabla\mathcal{H}(\mathbf{z}, t), \qquad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}. \tag{2.1}$$

where $\mathcal{H} : \mathbb{R}^{2n} \to \mathbb{R}$ denotes the Hamiltonian function and J is the $2n \times 2n$ canonical symplectic matrix, with I the $n \times n$ identity matrix. The vector $\mathbf{z} = (q, p)$ consists of canonically conjugate position variables $q \in \mathbb{R}^n$ and the corresponding momenta $p \in \mathbb{R}^n$. We write

$$\dot{q} = \frac{\partial\mathcal{H}}{\partial p}, \qquad \dot{p} = -\frac{\partial\mathcal{H}}{\partial q}, \tag{2.2}$$

where $\dot{q} = dq/dt$. These equations define a continuous-time dynamical system with phase space $z \in \mathbb{R}^{2n}$ [11]. The phase space contains the set of all possible states of the system, with

the initial condition $z_0 = (q_0, p_0)$. The system's time evolution is characterized by the time-t flow map $\Phi^t : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$, such that $z(t) = \Phi^t(z_0)$. A trajectory or orbit of a point mass in a dynamical system starting at z_0 is then defined by an ordered subset of the phase space z, as

$$\mathrm{Orb}(z_0) = \left\{ z \in \mathbb{R}^{2n} \ : \ z = \Phi^t(z_0),\ t \in I \subset \mathbb{R} \right\}. \tag{2.3}$$

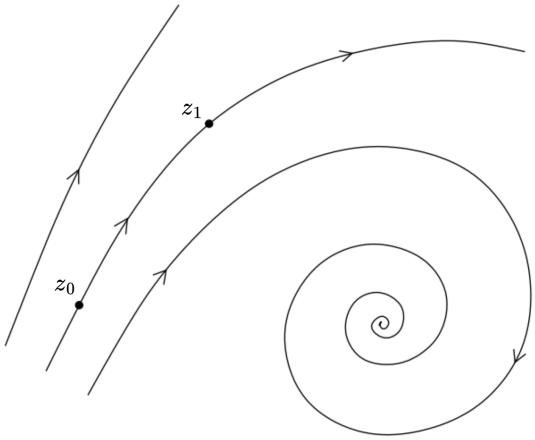

Figure 2.1: Visualization of an orbit in a dynamical system, modified from [11]

A particularly important class of Hamiltonian systems is that of *separable* Hamiltonians, for which the total energy can be written as

$$\mathcal{H}(q, p) = \mathcal{T}(p) + \mathcal{V}(q), \tag{2.4}$$

where $\mathcal{T}$ denotes the kinetic energy and $\mathcal{V}$ the potential energy of the system. Many mechanical systems, including gravitational N-body systems, fall into this category, and in such cases the Hamiltonian coincides with the total energy [10, 12, 7]. Hamiltonian systems possess several important structural properties, two of the most fundamental being conservation of energy and symplecticity of the flow.

Energy conservation For an autonomous Hamiltonian system, that is, one for which the Hamiltonian does not depend explicitly on time, equation (2.2) implies that the Hamiltonian is conserved along the trajectories generated by Hamilton's equations. Differentiating $\mathcal{H}(q(t), p(t))$ with respect to time yields

$$\frac{d\mathcal{H}}{dt} = \frac{\partial \mathcal{H}}{\partial q} \cdot \dot{q} + \frac{\partial \mathcal{H}}{\partial p} \cdot \dot{p} = 0. \tag{2.5}$$

which shows that the Hamiltonian value is constant along solutions of the equations of motion [7, p. 98]. In mechanical systems, this corresponds to conservation of the total energy.

Symplecticity of the flow Beyond energy conservation, Hamiltonian systems are characterized by a fundamental geometric property: the symplecticity of their flow map. The flow is symplectic if the Jacobian $D\Phi^t(z)$ satisfies

$$(D\Phi^t(z))^T J(D\Phi^t(z)) = J, \qquad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix},$$

Intuitively, symplecticity implies the preservation of the phase-space volumes and oriented area elements under the dynamics of Hamiltonian systems [13, p. 53]. Sets of the phase space z might change shapes as time t increases, but the volume they occupy in phase space remains the same [13, p. 55]. Symplecticity is crucial for long-time accuracy: numerical or learned models that fail to preserve this structure often exhibit artificial energy drift and qualitatively incorrect trajectories. Systems whose flow is symplectic admit a Hamiltonian formulation, making symplecticity a defining characteristic of Hamiltonian systems [7, Theorem 2.6].

2.1.2 The N-body problem

We now specialize the general Hamiltonian framework to the gravitational N-body problem, a model describing the motion of interacting point masses under Newtonian gravity. This system plays a central role in celestial mechanics and astrophysics and serves as a canonical example of a high-dimensional Hamiltonian system.

The Newtonian N-body problem describes the motion of N point masses $\{m_i\}_{i=1}^N$ interacting through pairwise gravitational forces [10]. Let $q_i(t) \in \mathbb{R}^3$ denote the position of the ith particle, $p_i(t) = \dot{q}_i(t)$ its velocity, and G the gravitational constant. In this work, we restrict attention to systems with unit masses, $m_i = 1$, in which case the momentum variables coincide with the velocities. The classical equations of motion follow from Newton's second law and law of gravity

$$\dot{q}_i = p_i, \qquad \dot{p}_i = -G \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \frac{q_i - q_j}{\|q_i - q_j\|^3}. \tag{2.6}$$

This system is conservative, with forces derived from a potential that depends only on the particle positions. As a consequence, the dynamics admit an equivalent Hamiltonian formulation. Defining the Hamiltonian

$$\mathcal{H}(q,p) = \underbrace{\sum_{i=1}^{N} \frac{\|p_i\|^2}{2m_i}}_{\mathcal{T}(p)} + \underbrace{\left(-G \sum_{1 \leq i < j \leq N} \frac{m_i m_j}{\|q_i - q_j\|} \right)}_{\mathcal{V}(q)}, \tag{2.7}$$

we observe that the Hamiltonian naturally decomposes into kinetic energy and potential energy. The Newtonian equations of motion given in (2.6) are recovered directly from Hamilton's equations by differentiating $\mathcal{H}$ with respect to the momentum p and position q, yielding

$$\dot{q}_i = \frac{\partial H}{\partial p_i} = \frac{p_i}{m_i} \overset{m_i=1}{=} p_i, \qquad \dot{p}_i = -\frac{\partial H}{\partial q_i} = -G \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{m_i m_j (q_i - q_j)}{\|q_i - q_j\|^3}. \tag{2.8}$$

This demonstrates the equivalence between the Hamiltonian and Newtonian formulations of the N-body problem. The system may also be written in compact form as shown in (2.1), which highlights the geometric structure of the phase space and the conservation of energy. The above equations are formulated for the spatial N-Body problem in $\mathbb{R}^3$ with $n = 3N$ degrees of freedom and phase space of dimension $\mathbb{R}^{6N}$. However, the same formulation applies directly to the planar N-body problem in $\mathbb{R}^2$, which will be simulated numerically in chapter 3.

2.1.3 The Two-body problem

The two-body problem, aside from the trivial one-body case, is the only instance of the gravitational N-body problem that is completely integrable. As such, it occupies a central position in classical mechanics and celestial dynamics [14, 15]. Given the forces as functions of the particle positions, the equations of motion uniquely determine the future evolution of the system from prescribed initial conditions, either analytically or numerically [15, p. 7].

Consider two point masses m_1 and m_2 and position vectors $q_1(t)$ and $q_2(t)$. Assuming a mutual attractive force depending only on the distance $q = q_2 - q_1$, the equations of motion are written as

$$\dot{p}_1 = -Gm_2 \frac{q_1 - q_2}{\|q\|^3}, \qquad \dot{p}_2 = -Gm_1 \frac{q_2 - q_1}{\|q\|^3}. \tag{2.9}$$

Introducing the barycenter (center of mass) of the system

$$q_{\text{cm}} = \frac{m_1 q_1 + m_2 q_2}{m_1 + m_2}, \tag{2.10}$$

one finds that $\ddot{q}_{\text{cm}} = 0$, so the center of mass moves uniformly along a straight line, and by subtracting the first equation in (2.9) from the second, the dynamics reduce to a single equation

$$\dot{p} = -G(m_1 + m_2) \frac{q}{\|q\|^3}, \tag{2.11}$$

which is equivalent to the equation of motion of a single particle attracted by a fixed center of mass $M = m_1 + m_2$. Thus, the two-body problem decomposes into the independent motion of the barycenter and a one-body central-force problem. This complete integrability makes the two-body case a natural baseline for validating HNNs before addressing chaotic systems such as the three-body problem. A representative example of the resulting bounded two-body motion and its associated energy evolution is shown in Figure 2.2.

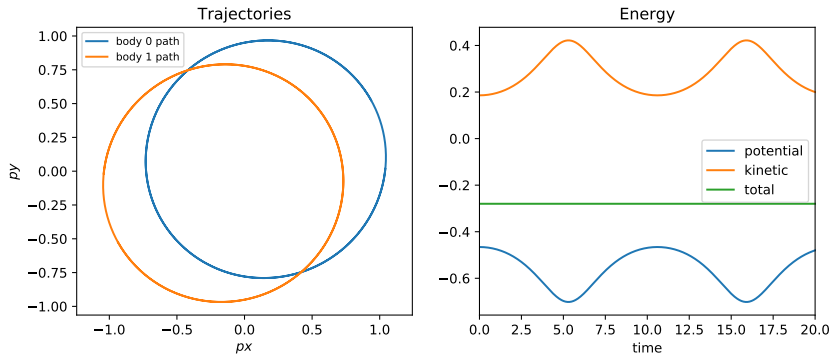

Figure 2.2: A representative two-body trajectory and its corresponding energy evolution are displayed. The left panel depicts stable motion about the center of mass, while the right panel illustrates the kinetic, potential, and total energies, showing energy conservation. Taken from [2, 16]

2.1.4 The General Three-body Problem

In contrast to the two-body problem, the general three-body problem represents the simplest gravitational system for which complete integrability is lost. More broadly, for $N > 2$, the gravitational N-body problem admits no general analytic solution in closed form. As discussed by Boccaletti et. al., the system of differential equations governing the three-body problem cannot be reduced to quadratures, and therefore the problem is not "solved" in the classical sense [14, Chapter 4]. The equations of motion

$$\dot{p}_1 = -Gm_1m_2\frac{q_1 - q_2}{\|q_1 - q_2\|^3} - Gm_1m_3\frac{q_1 - q_3}{\|q_1 - q_3\|^3}, \tag{2.12}$$

$$\dot{p}_2 = -Gm_2m_1\frac{q_2 - q_1}{\|q_2 - q_1\|^3} - Gm_2m_3\frac{q_2 - q_3}{\|q_2 - q_3\|^3}, \tag{2.13}$$

$$\dot{p}_3 = -Gm_3m_1\frac{q_3 - q_1}{\|q_3 - q_1\|^3} - Gm_3m_2\frac{q_3 - q_2}{\|q_3 - q_2\|^3}, \tag{2.14}$$

describe a system with phase space $z \in \mathbb{R}^{18}$, consisting of 18 first-order equations, which may be partially reduced using conserved quantities such as total energy, linear momentum, and angular momentum. However, as stated above, even after these reductions the resulting dynamics remain fundamentally non-integrable.

Despite this lack of a general solution, the three-body problem exhibits a remarkably rich structure. Although its behavior is chaotic, special initial conditions can give rise to stable, periodic, or quasi-periodic orbits, as illustrated in figures 2.3 and 2.4. These solutions occupy measure-zero subsets of phase space, but play an important conceptual and practical role in celestial mechanics. Well-known examples include choreographic solutions and symmetric periodic orbits, which persist due to delicate balances between gravitational forces and conserved quantities.

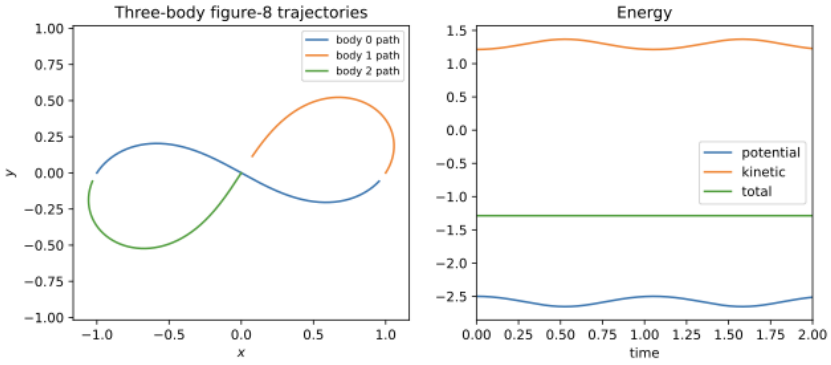

Figure 2.3: A Figure-8 stable periodic configuration [8] is displayed, in which all three bodies follow a single closed curve while conserving total energy.

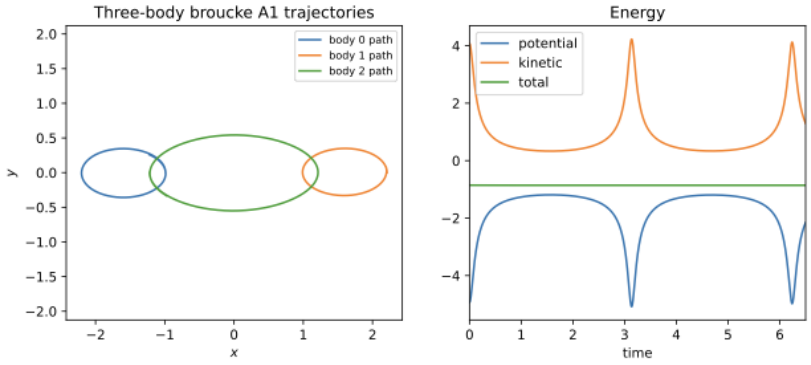

Figure 2.4: A Broucke A1 stable periodic configuration [9] is displayed, in which all three bodies follow a single closed curve while conserving total energy.

These examples demonstrate that, although the three-body problem is not integrable in general, stable solutions do exist for carefully chosen initial configurations. The coexistence of stable periodic motion and chaotic dynamics makes the three-body problem a canonical test case for Hamiltonian learning methods. Its sensitivity to initial conditions, coupled with strict conservation laws, provides a stringent benchmark for Hamiltonian Neural Networks, particularly in assessing their ability to preserve long-term qualitative structure while modeling complex nonlinear dynamics.

2.2 Numerical Integration for Hamiltonian Systems

Numerical integration provides a means of approximating the time evolution of a dynamical system. By discretizing time, the system's trajectory in phase space can be computed iteratively without requiring an explicit analytical solution of the equations of motion. We provide a brief introduction to numerical methods for approximating solutions of ordinary differential equations (ODEs), with particular focus on Hamiltonian systems governed by Hamilton's equations (2.1). For most Hamiltonian systems trajectories are not available in closed form. Consequently, numerical approximation is essential for simulating their dynamics [13]. A numerical integrator replaces the continuous flow $z(t) = \Phi^t(z_0)$ by a discrete sequence $\{z_n\}_{n\geq 0}$, where $z_n \approx z(t_n)$ at times $t_n = nh$ and $h > 0$ denotes the time step [7]. While many general-purpose integration schemes provide high local accuracy, their long-term behavior can differ substantially from that of the underlying Hamiltonian system. This section reviews classical integration methods and highlights the importance of structure-preserving, symplectic schemes for Hamiltonian dynamics.

2.2.1 Classical Time Integration Methods

Among the simplest numerical integrators for ODEs is the **explicit Euler** method, defined by

$$\begin{cases} q_{n+1} = q_n + h\nabla_p H(q_n, p_n), \\ p_{n+1} = p_n - h\nabla_q H(q_n, p_n), \end{cases} \tag{2.15}$$

given an initial condition $z_0 = (q_0, p_0)$. Although easy to implement, the explicit Euler method is only first-order accurate and is known to be unstable for many Hamiltonian systems, even over short time intervals [7, Chapter I].

Higher-order accuracy can be achieved using **Runge–Kutta** methods. A commonly used example is the classical fourth-order Runge–Kutta method (RK4), which attains fourth-order local accuracy by evaluating the vector field at multiple intermediate stages within each time step. Due to their high accuracy and robustness, Runge–Kutta methods are widely used in scientific computing and serve as a standard baseline for numerical simulation. However, despite their favorable short-term accuracy, such methods do not preserve the geometric structure of Hamiltonian systems. As a result, long-term integrations may suffer from *energy drift*, where numerical energy errors accumulate over time, even for high-order schemes such as RK4 [7, Chapters I and VI]. In conservative systems, where energy conservation

constrains the motion to invariant manifolds in phase space, numerical energy drift can cause trajectories to leave these manifolds, leading to artificial damping or excitation and fundamentally incorrect qualitative behavior.

2.2.2 Symplectic Integration Methods

To address these shortcomings, *symplectic integrators* have been developed specifically for Hamiltonian systems. A numerical method is symplectic if the discrete-time map it defines preserves the symplectic structure of phase space. Unlike general-purpose integrators, symplectic methods reproduce key geometric properties of the exact flow, including phase-space volume preservation and near-conservation of energy over long time horizons [7, Chapter VI].

Symplectic Euler Method. One of the simplest examples of a symplectic integrator is the symplectic Euler method, originally introduced by de Vogelaere [7]. It consists of two *first-order* schemes [7, Theorem 3.3], given by

$$\begin{cases} q_{n+1} = q_n + h\nabla_p H(p_{n+1}, q_n), \\ p_{n+1} = p_n - h\nabla_q H(p_{n+1}, q_n). \end{cases} \quad \text{or} \quad \begin{cases} q_{n+1} = q_n + h\nabla_p H(p_n, q_{n+1}), \\ p_{n+1} = p_n - h\nabla_q H(p_n, q_{n+1}). \end{cases} \tag{2.16}$$

For general Hamiltonian systems, these schemes are implicit, since the update of one variable depends on the unknown value of the other. For separable Hamiltonians however, both variants reduce to fully explicit update rules, because the gradients $\nabla_p H$ and $\nabla_q H$ depend only on p and q, respectively [7, Chapter I]. Despite being only first-order accurate, this method is symplectic and therefore preserves the qualitative structure of Hamiltonian dynamics. In particular, it avoids the systematic energy drift observed in the explicit Euler method and exhibits bounded energy errors over long time intervals [7, Chapter VI].

Implicit Midpoint Method. The implicit midpoint rule is another important symplectic integrator, defined for general Hamiltonian systems by

$$\begin{cases} q_{n+1} = q_n + h\,\nabla_p H\left(\frac{q_n+q_{n+1}}{2}, \frac{p_n+p_{n+1}}{2}\right), \\ p_{n+1} = p_n - h\,\nabla_q H\left(\frac{q_n+q_{n+1}}{2}, \frac{p_n+p_{n+1}}{2}\right). \end{cases} \tag{2.17}$$

This method is second-order accurate, time-reversible, and symplectic. Unlike explicit schemes, it requires the solution of an implicit equation at each time step, which increases computational cost. However, the implicit midpoint method preserves quadratic invariants exactly and exhibits excellent long-term stability properties [7, Chapter VI]. For this reason, it is frequently used as a reference method in the analysis of Hamiltonian integrators.

Störmer–Verlet Method. For separable Hamiltonians, one of the most widely used symplectic methods is the Störmer–Verlet integrator. It can be written as

$$\begin{cases} p_{n+\frac{1}{2}} = p_n - \frac{h}{2} \nabla_q H(p_n, q_n), \\ q_{n+1} = q_n + h \nabla_p H(p_{n+\frac{1}{2}}, q_n), \\ p_{n+1} = p_{n+\frac{1}{2}} - \frac{h}{2} \nabla_q H(p_{n+\frac{1}{2}}, q_{n+1}). \end{cases} \tag{2.18}$$

As with the implicit midpoint method, Störmer–Verlet is *second-order* accurate, symplectic, and time-reversible. Compared to the implicit midpoint method, Störmer–Verlet offers substantially lower computational cost, since it is fully explicit in this setting. Similarly to symplectic euler it is explicit only for separable Hamiltonians.

Backward error analysis shows that symplectic integrators conserve a nearby modified Hamiltonian over exponentially long times. For time-reversible schemes such as the implicit midpoint and Störmer–Verlet methods, this modified Hamiltonian is especially close to the original, leading to near-periodic energy errors, whereas the non-time-reversible symplectic Euler method exhibits a first-order deviation [7, Chapter IX].

The contrast between non-symplectic and symplectic integration methods highlights the importance of preserving geometric structure when modeling Hamiltonian systems. In data-driven settings, learned dynamical models that fail to respect invariants such as symplecticity are subject to the same long-term instabilities as non-symplectic numerical integrators. This observation motivates the development of structure-preserving learning approaches, such as Hamiltonian Neural Networks, which aim to encode Hamiltonian structure directly into the model.

2.3 Neural Networks for Dynamical Systems

Classical methods for dynamical systems, such as numerical solvers for ordinary differential equations, have proven robust across a wide range of applications. However, these methods assume that the governing equations and their structural form are known a priori. In settings where the underlying dynamics or physical laws are unknown or only partially specified, data-driven approaches based on neural networks provide an alternative by learning the system dynamics directly from observations.

Feed-Forward Neural Networks We briefly recall the definition of a feed-forward neural network (FFNN) in the context of regression [17], where the objective is to approximate a scalar-valued function such as the Hamiltonian $\mathcal{H}$. This class of networks will serve as a fundamental building block for the learning frameworks considered later. Let $X \subset \mathbb{R}^D$ denote the input space and $Y \subset \mathbb{R}$ the output space. A feed-forward neural network with L hidden layers defines a function $\Phi : X \to Y$, constructed as a composition of affine transformations and nonlinear activation functions.

Let $\Phi^{(l)}(x)$ define the output of the l-th layer for an input $x \in \mathbb{R}^D$, the network is then defined recursively as

$$\Phi^{(l)}(x) = \begin{cases} x, & \text{for } l = 0, \\ \sigma\Big(W_l \Phi^{(l-1)}(x) - b_l\Big), & \text{for } 0 < l \leq L, \\ W_l \Phi^{(l-1)}(x) - b_l, & \text{for } l = L+1, \end{cases} \tag{2.19}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ denotes a nonlinear activation function applied element-wise, and $\theta = \{W_l, b_l\}_{l=1}^{L+1}$ are the trainable parameters, which are the weights and biases of the network. The matrices $W_l \in \mathbb{R}^{N_l \times N_{l-1}}$ and vectors $b_l \in \mathbb{R}^{N_l}$ define the affine transformations at each layer, where N_l denotes the number of neurons in the l-th layer, with $N_0 = D$ and $N_{L+1} = 1$ for scalar-valued regression. The final layer is taken to be linear, which is standard in regression settings. Given a dataset $\{(x_i, y_i)\}_{i=1}^N \subset X \times Y$, the parameters θ are learned by minimizing an empirical loss functional of the form

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell\big(\Phi(x_i; \theta), y_i\big), \tag{2.20}$$

where $\ell : Y \times Y \to \mathbb{R}$ denotes a suitable pointwise loss function, such as the squared error. In the regression setting considered here, this corresponds to approximating an unknown target function from data byoptimizing the network parameters. The minimization of the empirical loss functional is typically carried out using gradient-based optimization methods. Gradients of the loss with respect to the network parameters are efficiently computed via the backpropagation algorithm, which applies the chain rule of calculus across the network layers. The parameters are then updated iteratively according to

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}\big(\theta^{(t)}\big),$$

where $\eta > 0$ denotes the learning rate. In practice, the gradient is often approximated using stochastic or mini-batch variants of gradient descent.

Neural ODEs (NODEs) While feed-forward neural networks model static input–output mappings, many physical systems are more naturally described by continuous-time dynamics. This observation motivates the use of neural networks to parametrize the vector fields of ordinary differential equations (ODEs), leading to the framework of *Neural Ordinary Differential Equations* (Neural ODEs) [18].

A key motivation for Neural ODEs stems from the interpretation of deep residual networks (ResNets) as discrete dynamical systems. Residual learning is particularly effective when the desired mapping is close to the identity, such as in image recognition tasks where features are progressively refined across layers, or in dynamical systems where successive states differ only by small incremental changes [19]. In a ResNet, rather than learning the full transformation of one hidden state h_k to the next h_{k+1}, the network only learns their residual, and updates the hidden state according to $h_{k+1} = h_k + F(h_k, \theta_k)$. This update rule is equivalent to a forward

Euler discretization with unit step size. As the step size decreases and the number of layers increases, this discrete update converges to the continuous-time formulation $\frac{dh(t)}{dt} = F(h(t), t)$, which corresponds to the definition of an ordinary differential equation.

Let $\{z_i\}_{i=0}^{T} \subset \mathbb{R}^d$ denote a trajectory of system states observed at uniformly spaced time points $\{t_i\}_{i=0}^{T}$. Many dynamical systems can be expressed as ordinary differential equations of the form $\frac{dz}{dt} = F(z)$, where z represents the system state and $F : \mathbb{R}^d \to \mathbb{R}^d$ defines the underlying vector field. When the dynamics are time-invariant, the function F specifies a flow on state space that determines the system's evolution from any initial condition. Neural ODE models approximate the right hand side (RHS) of this ODE, the vector field F, using a neural network F_θ with parameters θ.

$$\frac{dz}{dt} = F_\theta(z) \tag{2.21}$$

The objective is to infer F_θ from data such that the resulting continuous-time dynamics reproduce the observed trajectories. In Hamiltonian systems, the target ODE is $\frac{dz}{dt} = J\nabla H(z)$. Given an initial condition z_0, the model's predicted trajectory up to time T is defined as

$$\{\hat{z}_i(\theta)\}_{i=0}^{T} = \text{ODESolve}\Big(F_\theta,\, z_0,\, \{t_i\}_{i=0}^{T}\Big)\,,$$

where ODESolve uses any classical or symplectic integrator. Parameters θ can be learned by minimizing the discrepancy between predicted and observed trajectories. Therefore, we write the loss function as

$$\mathcal{L}_2(\theta) = \sum_{i=1}^{T} \|z_i - \hat{z}_i\|_2\,.$$

Unlike standard FFNNs, Neural ODEs have continuous depth and are trained using the adjoined sensitivity method instead of backprogagation, meaning that they do not require storing intermediate activations, making them memory efficient. Consequently, the model output is treated as the result of a black-box differential equation solver. Once trained, the learned vector field F_θ can be used to generate trajectories from previously unseen initial conditions. However, this approach presents significant limitations when modeling physical systems with conserved quantities or geometric structure. Although symplectic integrators may be used during numerical integration, the model does not learn the exact conservation laws, such as energy preservation, leading to accumulated errors and long-term drift in the learned dynamics. This motivates the incorporation of Hamiltonian structure into the model architecture.

2.4 Hamiltonian Neural Networks (HNNs)

Many physical systems are governed by conservation laws, such as total energy in Hamiltonian systems. Standard neural networks, including generic Neural ODEs, do not explicitly enforce these invariants, which can lead to energy drift and unstable long-term simulations. Several approaches have been proposed to incorporate physical priors into learning-based models. Physics-Informed Neural Networks (PINNs) introduce physical constraints through

additional terms in the loss function, penalizing violations of governing equations or conservation laws [1]. While PINNs incorporate physical knowledge only through the loss function, other methods additionally embed physical structure directly into the model architecture. Hamiltonian Neural Networks (HNNs) [2, 6], differ fundamentally from both PINNs and generic Neural ODEs. Given canonical input coordinates (q, p), instead of learning an unconstrained vector field, HNNs parametrize the Hamiltonian function and output a single energy like value $\mathcal{H}_\theta$. The time derivatives $\dot{q}$ and $\dot{p}$ are then recovered from $\mathcal{H}_\theta$ directly via differentiation, as shown in equation (2.2) and illustrated in Figure 2.5. In the following, we describe the basic architecture of Hamiltonian Neural Networks.

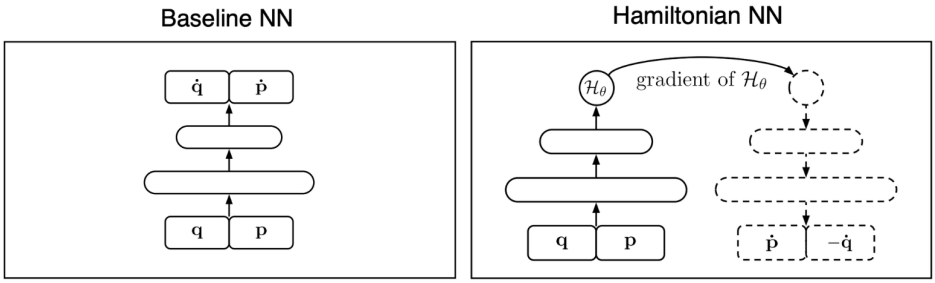

Figure 2.5: Comparison of neural dynamics models, taken from [2], is displayed. *Left:* A baseline neural network directly learns the time derivatives $(\dot{q}, \dot{p})$ from the state (q, p), corresponding to an unconstrained vector field. *Right:* A Hamiltonian Neural Network (HNN) learns a scalar Hamiltonian $H_\theta(q, p)$ and recovers the dynamics via Hamilton's equations

Hamiltonian Neural Networks are trained using supervision on the system's time derivatives, while remaining unsupervised with respect to the Hamiltonian itself. Rather than requiring energy labels, the model enforces consistency with Hamilton's equations by penalizing discrepancies between the observed time derivatives and those implied by the gradients of the learned Hamiltonian, as defined below

$$\mathcal{L}_{\text{HNN}} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial p} - \frac{dq}{dt} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial q} + \frac{dp}{dt} \right\|_2 . \tag{2.22}$$

Forward simulation of the learned dynamics is performed via numerical integration of the Hamiltonian vector field. In their experiments, Greydanus et al. demonstrate that a well-trained Hamiltonian Neural Network predicts the trajectories of N-body systems more accurately than a baseline neural dynamics model (Neural ODE) that learns the vector field directly [2]. Furthermore, their experiments yielded stable and accurate long-term predictions for the two-body problem, whereas performance degrades for the three-body problem, reflecting the increased difficulty of modeling chaotic dynamics. While Greydanus et al. employ a fourth-order Runge–Kutta (RK4) scheme in their experiments, the learned dynamics are Hamiltonian by construction, making symplectic integrators a natural choice. Symplectic methods better preserve the geometric structure of Hamiltonian systems and

typically exhibit improved long-term accuracy, particularly in chaotic systems. Subsequent work has shown that symplectic integrators lead to improved accuracy and generalization when applied to learned Hamiltonian models [4, 20, 6].

2.5 Random Feature HNNs

Training HNNs using gradient-based optimization can be computationally expensive, as it requires repeated backpropagation through the Hamiltonian and its gradients. Moreover, iterative gradient-descent-based methods often suffer from slow convergence rates in neural network training [21]. To mitigate this cost while preserving the structural inductive bias imposed by Hamilton's equations, recent work has proposed combining HNNs with random feature models, resulting in Random Feature Hamiltonian Neural Networks (RF-HNNs) [3]. The central idea is to avoid iterative optimization of all network parameters by fixing the hidden-layer weights and biases a priori. These hidden-layer parameters can be chosen either randomly or via data-driven sampling. Only the final linear layer is trained, which reduces learning to a convex least-squares problem, substantially lowering computational cost while retaining Hamiltonian structure in the learned vector field. A central challenge in RF-HNNs is therefore the choice of the probability distribution used to sample the hidden-layer parameters. In the following we differential two strategies.

Data-agnostic sampling A common data-agnostic strategy is the Extreme Learning Machine (ELM). For each hidden layer $l = 1, \dots, L$, weights and biases are sampled independently from a data-agnostic distribution, such as $W_l \sim \mathcal{N}(0, I)$, $b_l \sim \mathcal{U}[-1, 1]$, and remain fixed. The final layer is obtained by linear least squares.

Data-driven sampling Bolager et al. [5] proposed **Sample Where It Matters (SWIM)**, a data-driven random feature method that constructs weights and biases parameters directly from data, instead of sampling from a data-agnostic distribution. The core idea of SWIM is to place activation functions in regions of the input space where the target function exhibits large variations, such that the resulting random features are aligned with the gradients of the target function. The authors show that SWIM-sampled networks achieve consistently higher accuracy than ELMs across a range of tasks. Moreover, their performance is often comparable to that of fully trained neural networks, while maintaining orders-of-magnitude faster training times due to the absence of iterative gradient-based optimization. Let Φ be a FFNN with L hidden layers. For each layer l and neuron i, SWIM samples a pair of points $(x_{0,i}^{(1)}, x_{0,i}^{(2)}) \in \mathcal{X} \times \mathcal{X}$ and defines the corresponding weight and bias as

$$w_{l,i} = s_1 \frac{x_{l-1,i}^{(2)} - x_{l-1,i}^{(1)}}{\left\| x_{l-1,i}^{(2)} - x_{l-1,i}^{(1)} \right\|^2}, \qquad b_{l,i} = \langle w_{l,i},\, x_{l-1,i}^{(1)} \rangle + s_2. \tag{2.23}$$

The constants $s_1, s_2 \in \mathbb{R}$ control the placement of the activation function σ relative to the sampled point pair $(x^{(1)}, x^{(2)})$. For ReLU activations, we set $s_1 = 1$ and $s_2 = 0$, which

positions the activation threshold between the two points and yields linear interpolation in regression settings. For tanh activations, we choose $s_1 = 2s_2$ and $s_2 = \ln(3)/2$, resulting in symmetric activation values. For the final layer, we solve

$$(W_{L+1}, b_{L+1}) = \arg\min_{W,b} \mathcal{L}\big(W\,\Phi^{(L)}(\cdot) + b\big),$$

where $\mathcal{L}$ denotes the chosen loss function. Since the hidden-layer parameters are fixed, this optimization reduces to a linear least-squares problem. This optimazation is visualized in the left panel (S-MLP) in Fig. 2.6. The right panel (S-HNN) modifies only the last step, by parameterizing the Hamiltonian $\mathcal{H}$.

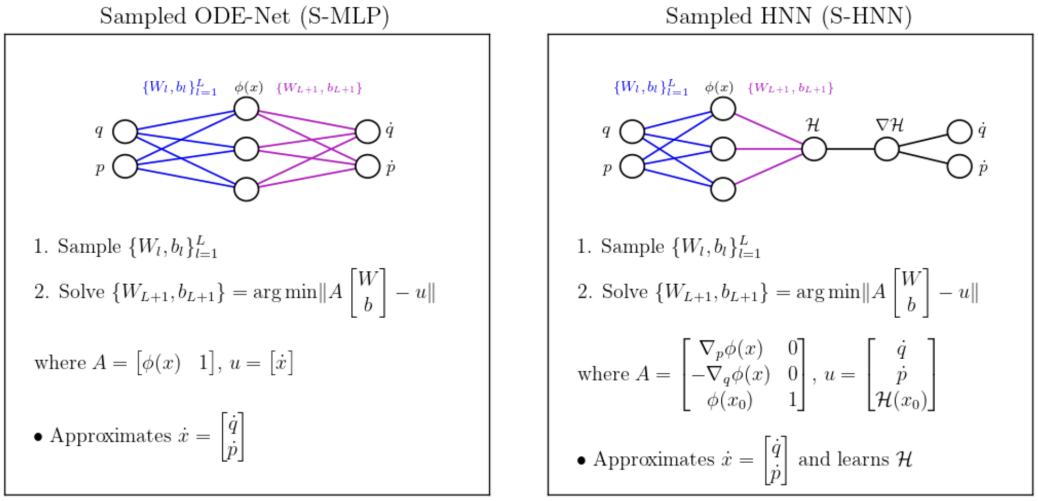

Figure 2.6: Comparison of S-MLP and S-HNN is displayed, taken from [3]

Depending on how the point pairs $(x^{(1)}, x^{(2)})$ are sampled, different variants of SWIM arise. In the fully supervised setting, where function values are available, SWIM samples point pairs according to a probability density proportional to a finite-difference approximation of the gradient magnitude, thereby concentrating neurons in regions where the target function varies rapidly [5]. In contrast, when no access to the true function values is available, pairs are sampled uniformly at random from the input space, resulting in the **Uniform-SWIM (U-SWIM)** variant. While U-SWIM does not exploit gradient information directly, it still makes use of data and is data-driven. To bridge the gap between supervised and unsupervised settings, Rahma et al. [3] introduce **Approximate-SWIM (A-SWIM)**. In A-SWIM, an initial approximation of the target function is first obtained using U-SWIM. The resulting approximate function values are then used in place of the true values to define the SWIM sampling density, and the hidden-layer parameters are resampled accordingly. This adaptive procedure allows A-SWIM to recover most of the accuracy benefits of supervised SWIM without requiring access to the true Hamiltonian values, at the cost of an additional resampling step.

3 Random Feature HNNs for N-Body Problems

In this chapter, we study the approximation of Hamiltonian functions for the gravitational two-body and three-body problems using Random Feature Hamiltonian Networks (RF-HNNs) and compare their performance against baseline models. Section 3.1 introduces the training setup, including the considered model classes, hyperparameters, and evaluation metrics. Section 3.2 presents the two-body experiments, detailing the orbit generation procedure (3.2.1), scaling experiment results (3.2.2) and evaluation of the simulated trajectories (3.2.3). Section 3.3 follows the same structure for the three-body problem, where the orbit generation procedure is presented first (3.3.1), followed by the next section where the scaling and trajectory experiments are divided into two case studies: the Figure-8 orbit (3.3.2–3.3.3) and the Broucke A1 orbit (3.3.4–3.3.5).

3.1 Training setup

As discussed in the previous chapter, Greydanus et al. demonstrated that Hamiltonian Neural Networks (HNNs) can learn accurate dynamics for the two-body problem, but performance degrades substantially in the chaotic three-body regime. In this chapter, our objective is to improve the prediction of the trajectory for the three-body system and to systematically compare the models listed in Table 3.1 across both the two-body and three-body settings.

Abbreviation	**Model Description**
MLP	Baseline Neural ODEs
HNN	Hamiltonian Neural Networks
S-MLP	Random Feature Neural ODEs
S-HNN	Random Feature Hamiltonian Networks

Table 3.1: Models compared in the two-body and three-body experiments are listed. The prefix "S-" denotes sampled (random-feature) models, and can be used interchangeably with the notation RF-MLP and RF-HNN.

We also investigated how different sampling strategies (SWIM, A-SWIM, U-SWIM, and ELM) for S-HNN and S-MLP affect approximation quality. In addition, whereas Greydanus et al. primarily employed a classical Runge–Kutta (RK4) solver for forward simulation, we use symplectic integrators (symplectic Euler and implicit midpoint), as these better respect the geometric structure of Hamiltonian systems. Finally, we make use of the training framework of Rahma et al., which was originally demonstrated on systems such as the double pendulum, Hénon–Heiles, Lotka–Volterra, and spring–mass dynamics [3]. We evaluate this framework on

gravitational two-body and three-body problems using an extended set of metrics, including relative training and test errors, scaling with network width and dataset size, simulations from unseen initial conditions, long-term energy behavior, and trajectory error growth relative to ground-truth integration. Extending the codebase of Rahma et. al. we define a new Hamiltonian for the two-body and three-body problems and initialize the model parameters with input dimension $4N$, corresponding to N bodies with two-dimensional positions and momenta. The MLP and HNN models are trained using a conventional gradient-based optimizer (Adam), whereas the sampled models (S-MLP and S-HNN) are trained via a linear least-squares solver induced by the random-feature parametrization.

Traditional training of MLP and HNN For torch-based models, implemented using PyTorch [22], training is performed via vector-field supervision. For an MLP-based ODE-Net, the network directly outputs the vector field $\hat{\dot{z}}_\theta(z) \in \mathbb{R}^d$, whereas the HNN internally outputs a scalar Hamiltonian $\hat{\mathcal{H}}_\theta(z)$, and returns the induced vector field through $\hat{\dot{z}}_\theta(q,p) = J\nabla\hat{\mathcal{H}}_\theta(q,p)$. The training objective is to minimize the mean squared error

$$\mathcal{L}(\theta) = \frac{1}{B}\sum_{i=1}^{B} \left\| \dot{z}_i - \hat{\dot{z}}_\theta(z_i) \right\|_2^2,$$

where B denotes the batch size. The model parameters θ are optimized using the Adam optimizer. The parameter updates are given by

$$\theta \leftarrow \mathrm{Adam}(\theta;\, \eta, \lambda),$$

where η denotes the learning rate and λ the weight decay coefficient. Training is performed for a fixed number of gradient updates. The optimization hyperparameters are summarized in Table 3.2.

Hyperparameter	**Value**
$\eta =$ `learning_rate`	10^{-3}
$\lambda =$ `weight_decay`	10^{-5}
`total_steps`	10^4

Table 3.2: Optimization hyperparameters for gradient-based training are listed.

Sampled training for S-MLP and S-HNN In sampled training, no gradient-based optimizer or learning rate is required. All output-layer parameters are obtained in closed form by solving a linear least-squares problem using a truncated solver with threshold `rcond`, which controls numerical stability by discarding near-singular directions. The sampling behavior is controlled by the choice of sampling strategy (ELM, SWIM, A-SWIM, U-SWIM). Unlike traditional training, sampled training does not use batch-wise training, instead the linear system is constructed from the full training set and solved in a single step. In this setting, ground-truth Hamiltonian values are additionally used, which are not employed in traditional

training. For SWIM-based sampled models, Hamiltonian values are used solely to guide the construction of hidden-layer random features by favoring feature placements in regions where the Hamiltonian varies strongly; they do not enter the least-squares optimization of the output-layer parameters.

Training evaluation metrics For the MLP and S-MLP models, performance is evaluated using the error in the predicted time derivatives. Specifically, we report the relative ℓ_2-norm error, given by

$$\text{error} = \frac{\|y_{\text{true}} - y_{\text{pred}}\|_2}{\|y_{\text{true}}\|_2} \tag{3.1}$$

For the HNN and S-HNN models, we additionally report two additional relative ℓ_2-norm errors: the error in the predicted Hamiltonian value and the error in the predicted Hamiltonian gradient. This allows us to assess not only vector-field accuracy, but also the quality with which the learned model captures the underlying Hamiltonian structure.

The codebase for all experiments is publicly available at

`https://github.com/sofiakloukianova/Sampling-HNNs`

3.2 Two-body system experiments

We consider the planar gravitational two-body problem as a first benchmark for learning Hamiltonian dynamics from data. Each body is assumed to have a unit mass and evolve under Newtonian gravity in two-dimensions with the gravitational constant set to $G = 1$. The system is described in section section 2.1 and has a phase-space state

$$z = (q_{x,1}, q_{y,1}, q_{x,2}, q_{y,2}, p_{x,1}, p_{y,1}, p_{x,2}, p_{y,2}) \in \mathbb{R}^8 \tag{3.2}$$

where $(q_{x,i}, q_{y,i})$ denote the x and y position coordinates of body i and $(p_{x,i}, p_{y,i})$ denote the corresponding momentum components, for $i \in 1, 2$. The Hamiltonian of the system is given by

$$H(q, p) = \sum_{i=1}^{2} \frac{1}{2}\|p_i\|^2 - \sum_{1 \le i < j \le 2} \frac{1}{\|q_i - q_j\|}, \tag{3.3}$$

as derived by equation 2.7 when setting $N = 2$, which induces the equations of motion through Hamilton's equations.

3.2.1 Orbit data generation

Training and test data are adapted from the setup of Greydanus et al. for the two-body system. They are generated by numerically integrating the true equations of motion from randomly sampled initial conditions corresponding to near-circular orbits. The numerical integrator evolves the system in two dimensions, where the state array has shape `(2, 5)` and contains masses `state[:,0]`, positions `state[:,1:3]`, and velocities `state[:,3:5]` for

each body. Orbits, as defined in equation (2.3), are generated by randomly sampling initial positions within a prescribed radial range. Specifically, the initial position of the first body is sampled as

$$q_1(0) \sim \text{Uniform}\big([q_{\min}, q_{\max}]^2\big), \qquad q_{\min} = 0.5,\ q_{\max} = 1.5.$$

The second body is initialized symmetrically with respect to the first, such that $q_2(0) = -q_1(0)$. A tangential velocity is selected to produce an approximately circular orbit, with a small multiplicative perturbation with orbit noise $\sigma_{\text{orbit}} = 5 \times 10^{-2}$ to induce mild ellipticity. Although Greydanus et al. use Gaussian *noise* $\sim \mathcal{N}\big(0, \sigma^2_{\text{orbit}}\big)$, we instead employ bounded uniform noise. Specifically,

$$p_1(0) \approx \frac{1}{2\|q_1(0)\|^{3/2}} \begin{pmatrix} -y_1(0) \\ x_1(0) \end{pmatrix} \big(1 + \text{noise}\big),$$

where

$$\textit{noise} \sim \text{Uniform}(-1, 1)\sigma_{\text{orbit}}.$$

This ensures controlled perturbations of the initial conditions and avoids rare large deviations that can arise from unbounded distributions. The second body is initialized symmetrically such that the center-of-mass frame is preserved: $p_2(0) = -p_1(0)$. For each initial condition, we now integrate over the time interval $t \in [t_0, t_1]$ in equally spaced evaluation time steps t_k with $k = 0, ..., T-1$, to obtain the resulting orbit $Orb(z_0) \in \mathbb{R}^{2\times 5}$ with shape `(2, 5, T)`. Since the learning data set does not include masses, we convert each integrated state into a vector as defined in (3.2). For each state z, we compute

$$\dot{z} = \frac{d}{dt} z(t) = \big(\dot{x}_1, \dot{y}_1, \dot{x}_2, \dot{y}_2, \dot{v}_{x,1}, \dot{v}_{y,1}, \dot{v}_{x,2}, \dot{v}_{y,2}\big) \in \mathbb{R}^8 \tag{3.4}$$

analytically from the governing equations of motion, which makes the ground truth vector field values. We also compute the kinetic and potential energy of the state and add them to obtain the total energy labels $H(z_i)$. Each data point also stores the exact Hamiltonian gradient $\nabla H(z_i)$. The final data set therefore consists of

$$(z_i,\ \dot{z}_i,\ H(z_i),\ \nabla H(z_i)).$$

From the complete set of generated orbit data, we construct an 80/20 split into training and test sets.

3.2.2 Dataset size and network width scaling

Using the generated orbit data as described in subsection 3.2.1 and the training setup in section 3.1, we evaluate all four models listed in Table 3.1 using two experimental settings: scaling the network width and scaling the training dataset size. In both cases, performance is evaluated using the relative ℓ_2 test error in the predicted time derivatives.

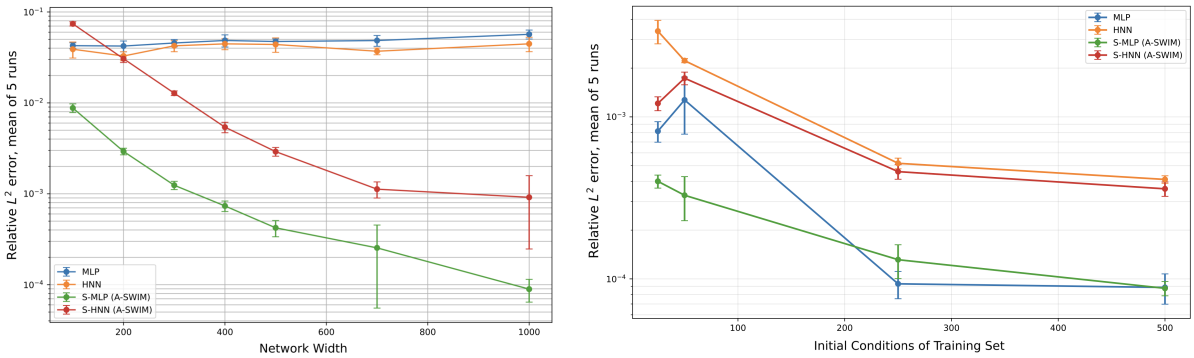

Figure 3.1: Results of the two-body scaling experiments are shown, measuring relative ℓ_2 test error (mean $\pm$ standard deviation over 5 seeds). *Left*: Test error is plotted against network width, using fixed train size 10000. Right: Test error is plotted against train size, using width 1000.

Network width scaling experiment To study the effect of model capacity, we fixed the training set size to 10000 and the test set size to 2000 and varied the network width. Specifically, models are trained with widths $\{100, 200, 300, 400, 500, 700, 1000\}$. For each width, the model is trained from scratch and evaluated on the same test set. All results are reported as averages over five independent runs with different random seeds. The results shown in the left panel of Figure 3.1 show no improvement in the performance of traditionally trained models with increasing width, while sampled models exhibit a pronounced improvement as width increases. This behavior may be attributed to the slower convergence of gradient-based training for the traditionally trained models compared to the sampled models, as well as to suboptimal hyperparameter tuning of traditionally trained models.

Dataset size scaling experiment To analyze data efficiency, we fixed the network width at 1000 and varied the number of training samples. The sizes of the training set are chosen as $\{500, 1000, 5000, 10000\}$, however, the number of time steps is kept constant, and only the number of initial conditions is increased, corresponding to the initial conditions $\{25, 50, 250, 500\}$. A single large training pool is generated once, randomly shuffled, and progressively subsampled to obtain the desired data set sizes, ensuring fair comparisons across conditions. Each model is retrained for every data set size and evaluated on the same fixed test set. The results shown in the right panel of Figure 3.1 indicate that all methods benefit from increased training data. However, both MLP-based models achieve lower errors than the HNN-based models. This can be attributed to their task of vector-field regression, for which MLPs offer greater flexibility and can fit the data more quickly, whereas HNNs are more constrained by the imposed Hamiltonian structure.

At this stage, the results do not show a clear advantage of the HNN or S-HNN models over the baseline MLP models in terms of relative test error. However, training and test accuracy alone is not sufficient to assess the quality of long-term trajectory prediction or energy conservation. As demonstrated in subsequent experiments, the Hamiltonian-based

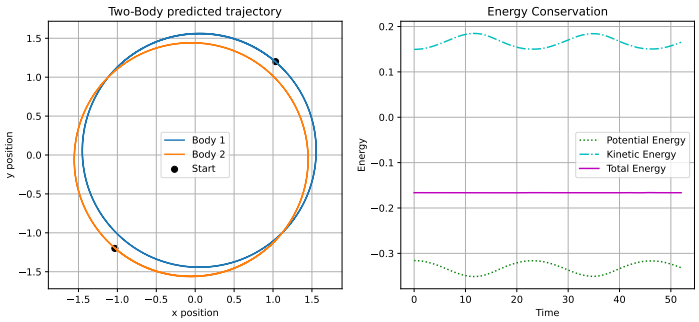

Figure 3.2: Predicted two-body trajectories (left) generated by S-HNN (A-SWIM), trained with train size 10000 and width 1000 and integrated via implicit midpoint with $\Delta t = 0.001$ for 50000 time steps are shown. Kinetic, potential energy and total energy are plotted over time (right).

models achieve substantially improved long-term trajectory stability and superior energy conservation behavior for the two-body problem simulations.

3.2.3 Trajectory evaluation

After training an S-HNN model with width 1000 and train-size 10000, and obtaining the training and test errors shown in Table 3.3, we proceed to evaluate the model in a forward simulation using the implicit-midpoint method over a long time horizon. This experiment assesses whether accurate local predictions of the Hamiltonian translate into stable long-term dynamics.

Target	Train Error	Test Error
$\dot{x}$	3.6749×10^{-3}	3.5663×10^{-3}
$H(x)$	7.7628×10^{-4}	7.7476×10^{-4}
$\nabla H(x)$	3.6749×10^{-3}	3.5663×10^{-3}

Table 3.3: Training and test relative L2 errors.

Figure 3.2 shows the two-body trajectory predicted by the S-HNN model with A-SWIM over 50000 time steps using $\Delta t = 0.001$. The predicted orbit remains close to a near-circular trajectory (left). The corresponding energy evolution (right) indicates that, although the kinetic and potential energies oscillate, the total energy is well conserved over time. In Figure 3.3, the predicted trajectory (middle) also closely matches the ground-truth orbit (left). Comparing the Hamiltonian values (right), the Hamiltonian along the predicted trajectory oscillates closely around the ground-truth value, and the model's learned Hamiltonian matches it even more closely, indicating that the S-HNN has successfully captured the underlying Hamiltonian structure in this simulation.

Next, we compare this result with the S-MLP model, which achieved lower test errors in

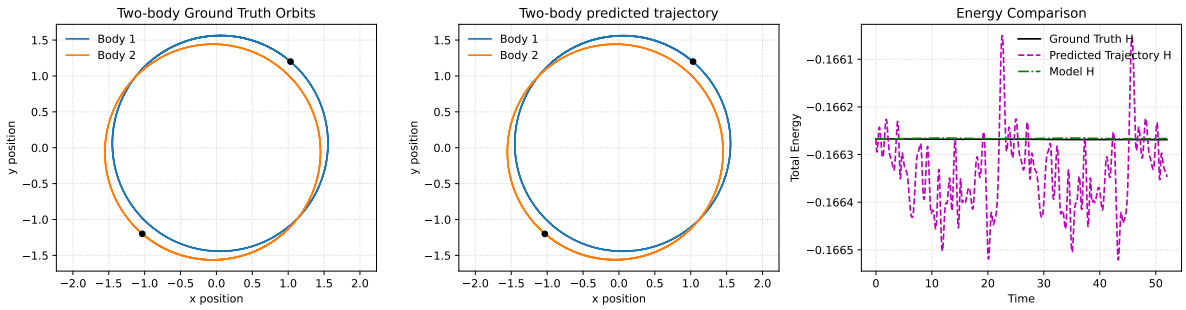

Figure 3.3: Ground-truth (left) and predicted two-body trajectories (middle), with total energy comparison (right) are shown for the S-HNN (A-SWIM) model.

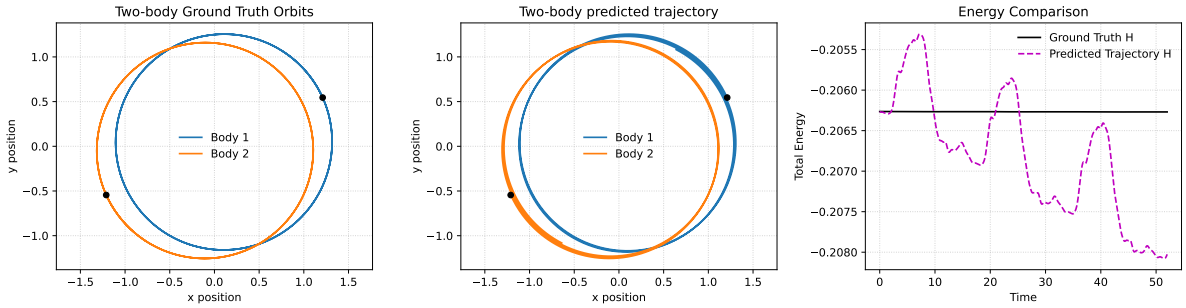

Figure 3.4: Ground-truth (left) and predicted two-body trajectories (middle), with total energy comparison (right) are shown for the S-MLP (A-SWIM) model.

the previous scaling experiments. In Figure 3.4, the baseline S-MLP shows a noticeable drift in the predicted trajectory over time and diverges from the ground-truth solution. The total energy also shows a clear drift, indicating a violation of the underlying Hamiltonian structure. In contrast, the S-HNN model preserves total energy much more accurately and produces trajectories that remain close to the ground-truth orbit over significantly long time horizons.

We further compare the sampled S-HNN model (Figure 3.3) with the traditionally trained HNN (Figure 3.5). The simulation results are consistent with the two-body training errors, where S-HNN achieved lower test error than HNN. While the HNN's predicted trajectories do not exhibit strong divergence in the Hamiltonian, the S-HNN model produces more stable orbit simulations and energy conservation. Nevertheless, the HNN still outperforms the baseline MLP, as illustrated in Figure 3.6.

3.3 Three-body system experiments

We consider the planar gravitational three-body problem as a challenging benchmark for learning Hamiltonian dynamics from data due to its chaotic behavior. As in the two-body case, each body is assumed to have unit mass and to evolve under Newtonian gravity in two

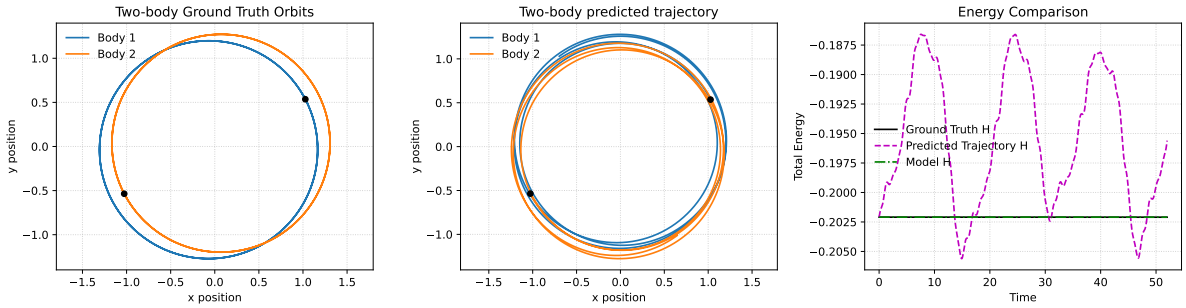

Figure 3.5: Ground-truth (left) and predicted two-body trajectories (middle), with total energy comparison (right) are shown for the HNN model.

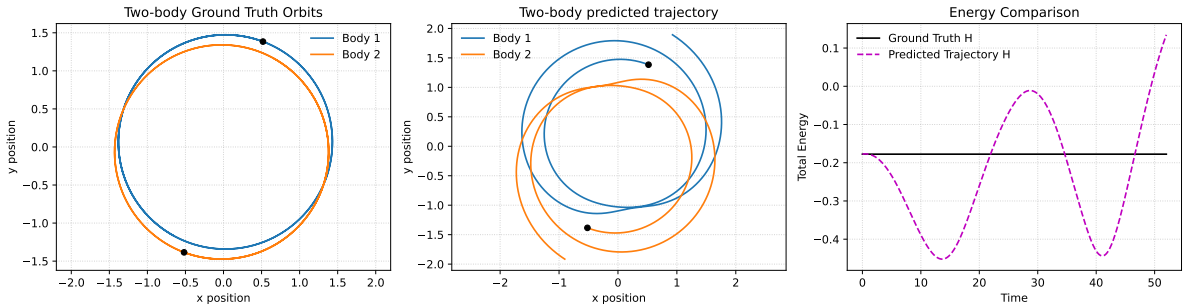

Figure 3.6: Ground-truth (left) and predicted two-body trajectories (middle), with total energy comparison (right) are shown for the MLP model.

dimensions with gravitational constant set to $G = 1$. The system has a phase-space state

$$z = (q_{x,1}, q_{y,1}, q_{x,2}, q_{y,2}, q_{x,3}, q_{y,3}, p_{x,1}, p_{y,1}, p_{x,2}, p_{y,2}, p_{x,3}, p_{y,3}) \in \mathbb{R}^{12},$$

where $(q_{x,i}, q_{y,i})$ denote the x and y position coordinates of body i and $(p_{x,i}, p_{y,i})$ denote the corresponding momentum components, for $i \in \{1, 2, 3\}$. The Hamiltonian of the system is given by

$$H(q, p) = \sum_{i=1}^{3} \frac{1}{2} \|p_i\|^2 \; - \sum_{1 \le i < j \le 3} \frac{1}{\|q_i - q_j\|},$$

which follows from the general Hamiltonian of the N-body problem for $N = 3$, and induces the equations of motion that follow from Hamilton's equations. We focus on two classes of stable periodic solutions as initial conditions: the Chenciner–Montgomery Figure-8 orbit and the Broucke A1 orbit (illustrated in Figure 2.3 and Figure 2.4).

3.3.1 Orbit data generation

To construct training and test datasets for the three-body problem, we start from known periodic solutions of the Newtonian three-body system: the Chenciner–Montgomery Figure-8 orbit and the Broucke A1 orbit. The initial positions and velocities for these trajectories are taken from published reference values [8, 9]. Each initial condition is stored in the format: $state \in \mathbb{R}^{3\times 5}$ with $state_i = [m_i,\ q_{x,i},\ q_{x,i},\ p_{x,i},\ p_{y,i}]$,, and then integrated forward in time using a symplectic integrator to obtain a high-resolution base periodic orbit. From this reference orbit, we select a set of initial conditions uniformly spaced along the trajectory, as illustrated in Figure 3.7. To enrich the data set, we perturb the sampled initial conditions. Similarly to the two-body data generation procedure, we add uniform noise to the position variable q. This produces a set of initial conditions that remain close to the stable periodic orbits but are no longer stable and diverge over time. For each perturbed initial condition, we integrate the system over a fixed time interval t and record samples of these trajectories. As in the two-body experiments, the trajectories are converted to canonical coordinates $z = (q, p) \in \mathbb{R}^{12}$ and labeled with ground-truth derivatives $\dot{z}$, Hamiltonian values $H(z)$ and Hamiltonian gradients $\nabla H(z)$ obtained from the analytical solution.

3.3.2 Figure-8 Dataset Size and Network Width Scaling

We conduct the same scaling experiments for the Figure-8 system as in the two-body case, examining the effects of network width and dataset size on model performance.

Network width experiments Here, we repeat the network width experiment using the same range of widths as in Section 3.2.2. As shown in the left panel of Figure 3.8, increasing the network width consistently improves the performance of sampled models, while traditionally trained models exhibit little to no improvement, mirroring the behavior observed in the two-body experiments. The right panel compares different sampling strategies for the S-HNN model and indicates that data-driven sampling strategies achieve lower errors than data-agnostic approaches.

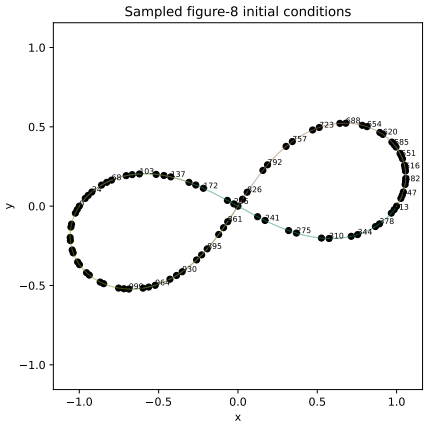

Figure 3.7: Sampling of Figure-8 initial conditions for the Figure-8 data generation is illustrated.

Dataset scaling experiment We repeat the data set scaling experiment as in the two-body case, but instead using training set sizes of $\{500, 1000, 5000, 10000, 15000\}$. As shown in Figure 3.9, sampled models achieve lower errors than traditionally trained models, consistent with the results observed for the two-body system. However, what is surprising here is that increasing the number of training initial conditions yields only limited improvement in performance. This behavior can be explained in part by the data generation procedure, since each perturbed initial condition produces trajectories that already cover a large portion of the Figure-8 region of phase space. In addition, the chaotic nature of the three-body dynamics likely also makes the system more difficult to learn accurately.

3.3.3 Figure-8 Trajectory evaluation

We evaluate forward trajectory prediction for the chaotic three-body system using the Figure-8 orbit. For trajectory simulation, we used initial conditions from the stable periodic solution without added noise. Despite the strong sensitivity of the three-body problem to the pertrubed initial conditions in the training data, learning the Hamiltonian structure remains beneficial for the Figure-8 case, as demonstrated below.

We also compare the performance of the symplectic Euler and the second-order implicit midpoint integrators using the S-HNN model with A-SWIM. As shown in Figure 3.10, the model approximately reproduces the orbit structure when integrated with symplectic Euler but diverges noticeably after some time. In contrast, the Figure also shows that the implicit midpoint method preserves the orbit structure significantly better over 7000 time steps. Nevertheless, long-term energy conservation cannot be guaranteed, and in our experiments, energy drift becomes noticeable after approximately 14000 time steps, due to the chaotic nature of the system.

The S-HNN model with implicit midpoint integration is further compared against the

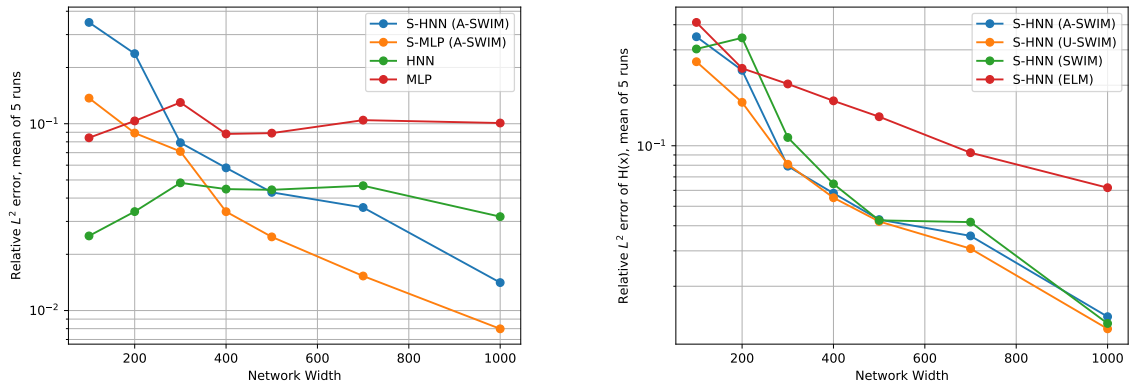

Figure 3.8: Relative test errors for the three-body figure-8 system are plotted against network width, for fixed training size 10000. Left: Comparison between traditionally trained and sampled models is shown. Right: Comparison of different sampling strategies for S-HNN is shown.

other model classes, again demonstrating superior performance. The energy evolution for the S-HNN is shown in Figure 3.11. Although larger deviations are observed than in the two-body case, the total energy remains approximately conserved over 7000 time steps, exhibiting no systematic drift relative to the ground-truth Hamiltonian. In contrast, Figure 3.12 shows that the S-MLP model exhibits a rapid energy spike for the 7000 time steps. For traditionally trained models, shown in Figures 3.13 and 3.14, this behavior is even more pronounced. Although the HNN conserves energy better than the baseline MLP, both are outperformed by the sampled S-HNN model in terms of long-term energy stability.

To compare across all four models, the energy trajectories are overlaid in a single plot. Figure 3.15 shows that the S-HNN better conserves energy than the S-MLP (left), while among traditionally trained models, the MLP exhibits faster divergence than the HNN (right).

To further measure trajectory divergence from the ground-truth orbits, the mean squared position error is evaluated over time for all four models. Figure 3.16 shows that the S-HNN achieves higher accuracy than the S-MLP, while the HNN outperforms the MLP.

3.3.4 Broucke A1 Dataset size and network width scaling

The Broucke A1 orbit represents a particularly challenging case in which all considered models fail to achieve accurate learning. Compared to the Figure-8 orbit, the Broucke A1 solution is dynamically less stable and exhibits strong momentum variations as well as pronounced spikes in both kinetic and potential energy. As a result, the relative test errors remain high across all model classes.

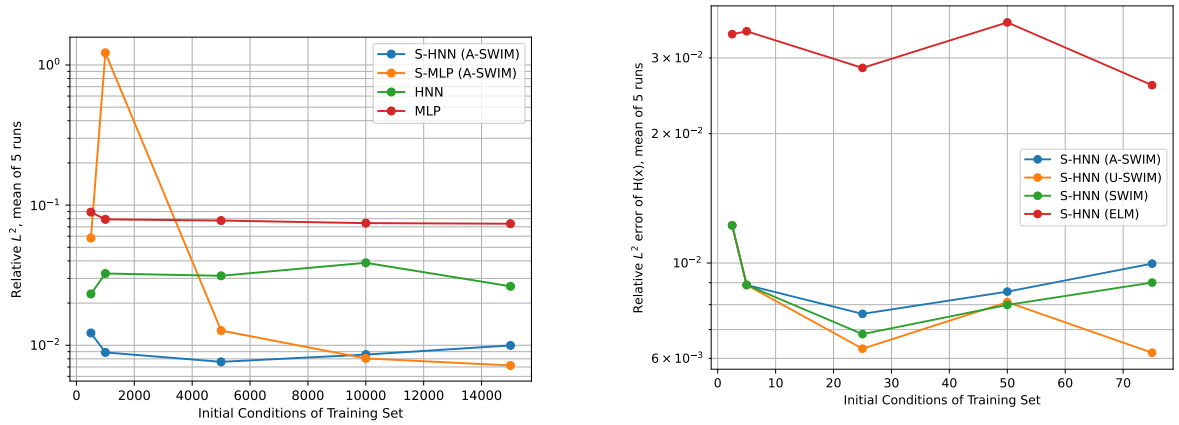

Figure 3.9: Relative test errors for the three-body figure-8 system are plotted against train size, for fixed width 1000. Left: Comparison between traditionally trained and sampled models is show. Right: Comparison of different sampling strategies for S-HNN is shown.

3.3.5 Broucke A1 Trajectory evaluation

These findings are reflected in the forward trajectory simulations. None of the models considered can produce accurate long-term predictions for the Broucke A1 orbit. For illustration, we present results up to 300 time steps for the S-HNN (A-SWIM) model and 700 time steps for the S-MLP (A-SWIM) model, beyond which all predicted trajectories diverge rapidly from the ground truth, and the remaining models exhibit similarly poor behavior.

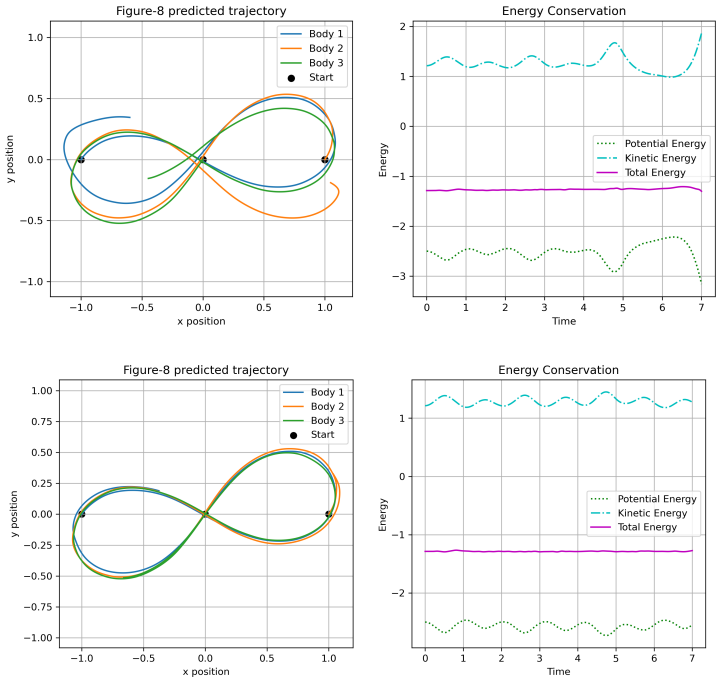

Figure 3.10: Predicted figure-8 three-body trajectory (left) generated by the S-HNN model (A-SWIN) using symplectic euler (upper row) and implicit midpoint (lower row) integration using $\Delta t = 0.001$ for 7000 steps, and corresponding energy evolution over time (right).

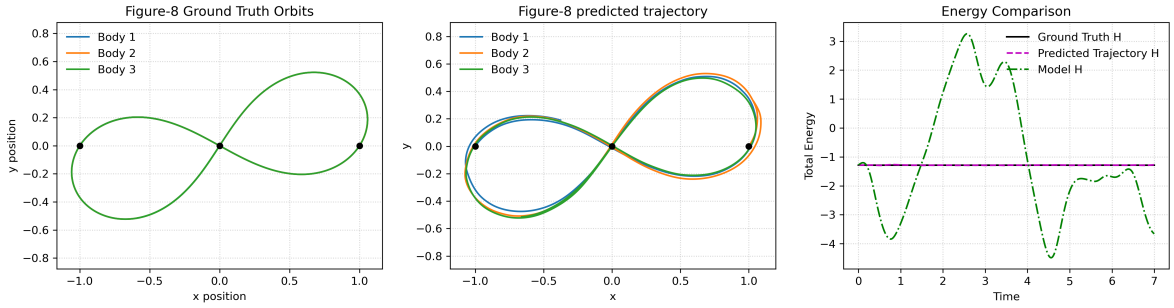

Figure 3.11: Ground-truth periodic orbits (left) and the predicted trajectory using implicit midpoint (middle) are compared. The Hamiltonian values for the ground truth, predicted trajectory, and S-HNN model are shown over 7000 time steps (right)

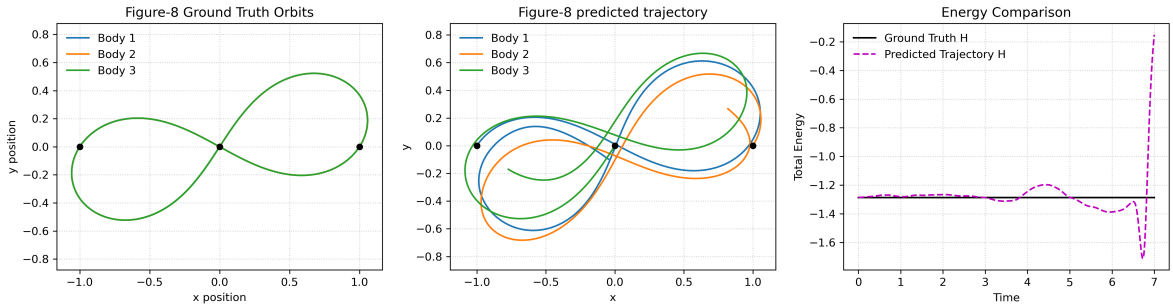

Figure 3.12: Ground-truth periodic orbits (left) and the predicted trajectory using implicit midpoint (middle) are compared. The Hamiltonian values for the ground truth, predicted trajectory, and S-MLP model are shown over 7000 time steps (right)

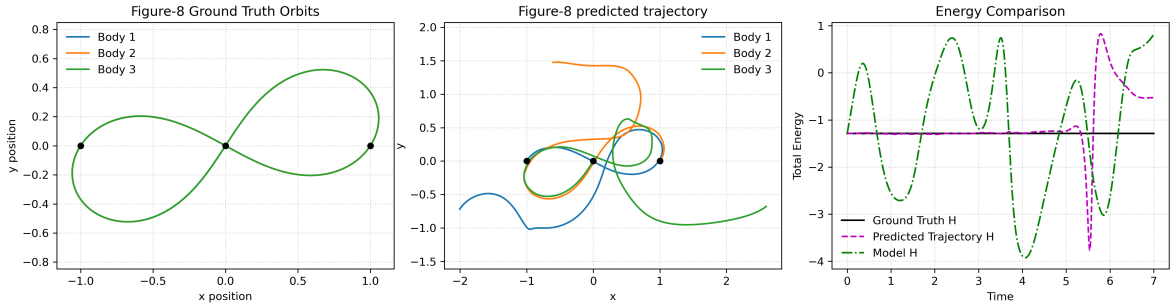

Figure 3.13: Ground-truth periodic orbits (left) and the predicted trajectory using implicit midpoint (middle) are compared. The Hamiltonian values for the ground truth, predicted trajectory, and HNN model are shown over 7000 time steps (right)

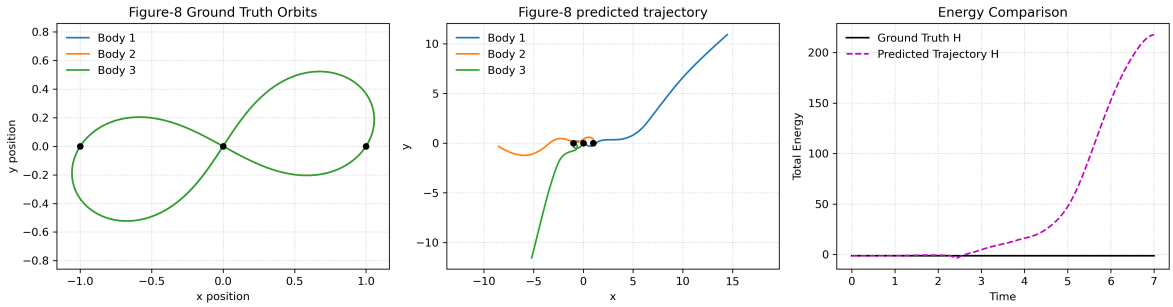

Figure 3.14: Ground-truth periodic orbits (left) and the predicted trajectory using implicit midpoint (middle) are compared. The Hamiltonian values for the ground truth, predicted trajectory, and MLP model are shown over 7000 time steps (right)

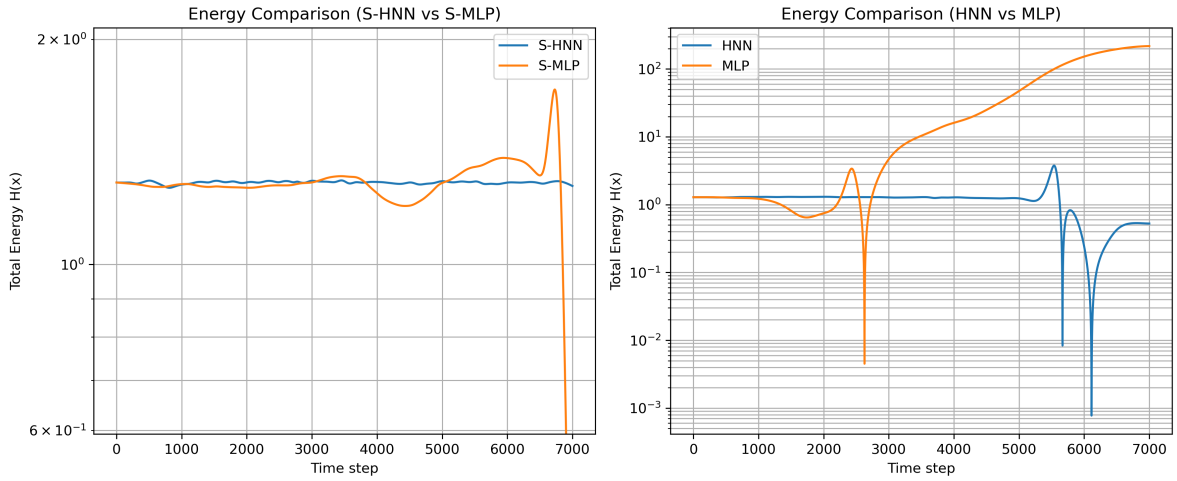

Figure 3.15: Total energy over time for three-body figure-8 trajectory simulations, comparing sampled models (S-HNN vs S-MLP, left) and traditionally trained models (HNN vs MLP, right).

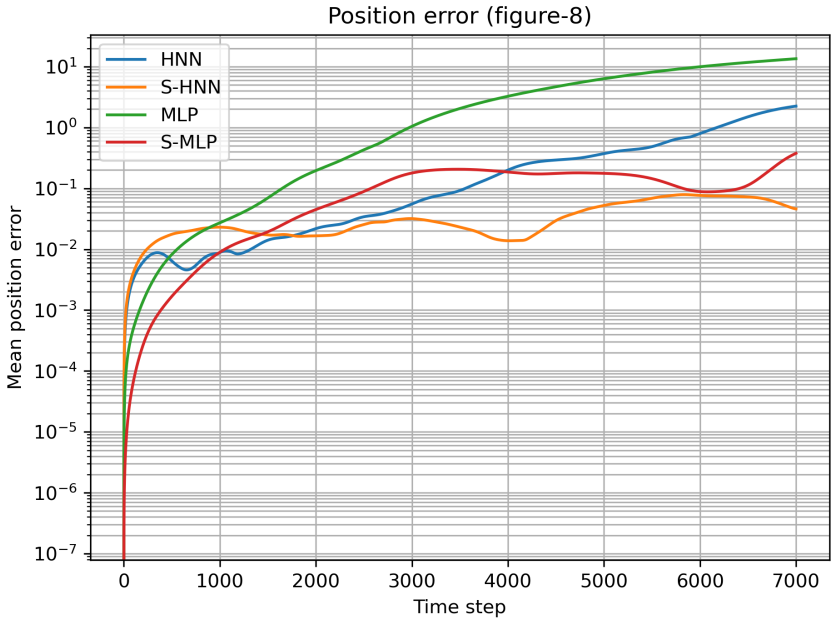

Figure 3.16: Mean squared position error over time for the three-body figure-8 trajectory is shown, comparing HNN, S-HNN, MLP, and S-MLP models.

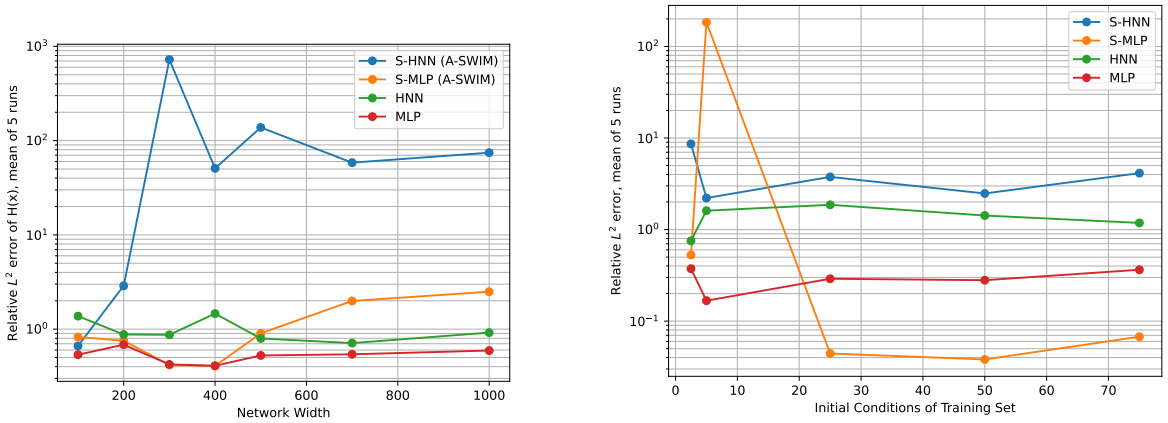

Figure 3.17: Relative test errors for the three-body Broucke A1 system plotted against network width (left) and training dataset size (right) for both sampled and traditionally trained models.

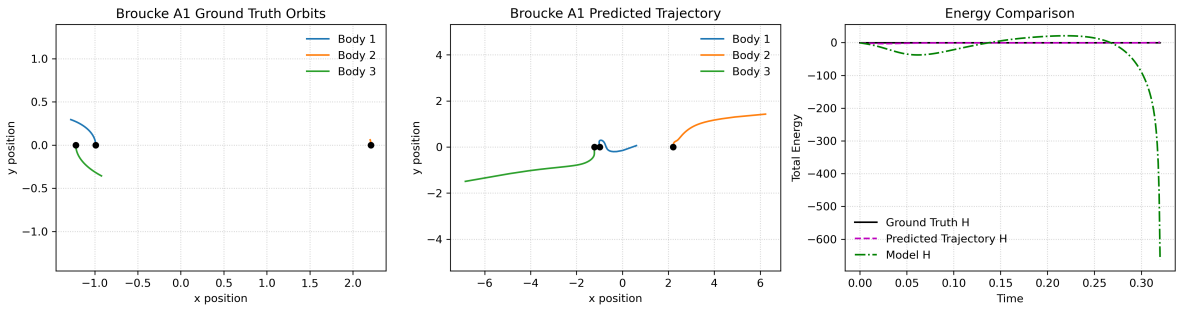

Figure 3.18: Ground-truth periodic orbit shown on the left compared to predicted trajectory of Broucke A1 in the middle. The right plot shows energy comparison between the ground truth, predicted orbit's and S-HNN model's Hamiltonian over 300 time steps.

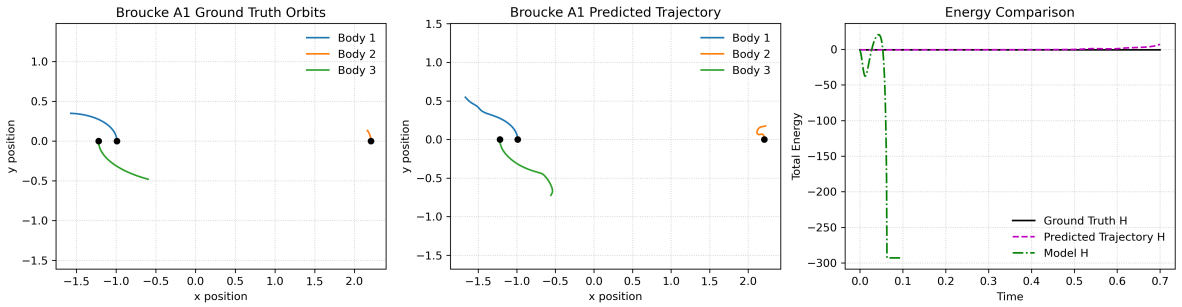

Figure 3.19: Ground-truth periodic orbit shown on the left compared to predicted trajectory of Broucke A1 in the middle. The right plot shows energy comparison between the ground truth, predicted orbit's and S-MLP model's Hamiltonian over 700 time steps.

4 Conclusion

This thesis evaluated Random Feature Hamiltonian Networks as structure-preserving, data-driven models for gravitational dynamics. Building on the training framework of Rahma et al. [3], the approach was extended to planar two- and three-body problems by implementing the corresponding Hamiltonians and constructing datasets with ground-truth derivatives, Hamiltonian values and Hamiltonian gradients. A systematic benchmark compared four model classes (MLP, HNN, S-MLP, S-HNN), multiple sampling strategies (ELM, SWIM, A-SWIM, U-SWIM), and symplectic integration schemes, using relative test errors, scaling with width and dataset size, long-horizon rollouts, and energy and trajectory error over time. The sampled models benefited greatly from increased network width, whereas the traditionally trained models showed little improvement under the same scaling. In the two-body system, test errors were not decisive, but trajectory-based evaluation favored the Hamiltonian structure: S-HNN achieved the most stable rollouts and energy conservation. In the three-body settings, increasing the number of training initial conditions led to only limited improvements in test error. A possible remedy would be to increase the diversity of the training data, for example by using stronger perturbations of initial conditions or by simulating trajectories over longer time horizons. For the Figure-8 orbit, S-HNN with implicit-midpoint preserved the qualitative orbit and energy behavior for substantially longer horizons than the baselines. However, the long-term accuracy degraded over time as a result of the chaotic nature of the three-body dynamics. Unlike the integrable two-body system, the three-body problem exhibits strong sensitivity to initial conditions and explores a higher-dimensional phase space, which makes learning accurate global dynamics significantly more difficult. This difficulty was further amplified for the Broucke A1 orbit, which is dynamically less stable and exhibits sharp variations in momentum and energy. In this regime, all models failed to achieve low test error or meaningful long-term trajectory predictions.

A natural extension of this work is the introduction of a separable sampled Hamiltonian model (S-HNN-SEP), in which the Hamiltonian is explicitly decomposed into kinetic and potential energy terms. Such a formulation would allow direct modeling of $T(p)$ and $V(q)$, enable the use of additional symplectic integrators such as Störmer–Verlet, which are explicit for separable Hamiltonians and allow for more detailed diagnostics of learned physical structure. In the current framework, only the total energy can be evaluated directly from the model, whereas a separable formulation would allow learned kinetic and potential energies to be analyzed independently. Beyond S-HNNs, other structured architectures could be explored. Random Feature Hamiltonian Graph Networks (RF-HGN) may better exploit interaction structure in many-body systems by explicitly encoding pairwise forces [23]. Symplectic Recurrent Neural Networks (SRNNs) could improve long-term stability through recurrence and implicit time-stepping [4]. These models may offer advantages in scalability,

interpretability, and robustness to noise, particularly for larger N-body systems, and would be valuable points of comparison for future work.

Bibliography

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "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 (2019), pp. 686–707.

[2] S. Greydanus, M. Dzamba, and J. Yosinski. "Hamiltonian neural networks". In: *Advances in neural information processing systems* 32 (2019).

[3] A. Rahma, C. Datar, and F. Dietrich. "Training Hamiltonian neural networks without backpropagation". In: *arXiv preprint arXiv:2411.17511* (2024).

[4] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. "Symplectic recurrent neural networks". In: *arXiv preprint arXiv:1909.13334* (2019).

[5] E. L. Bolager, I. Burak, C. Datar, Q. Sun, and F. Dietrich. "Sampling weights of deep neural networks". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 63075–63116.

[6] T. Bertalan, F. Dietrich, I. Mezić, and I. G. Kevrekidis. "On learning Hamiltonian systems from data". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.12 (2019).

[7] C. L. Ernst Hairer Gerhard Wanner. *Geometric Numerical Integration*. Vol. 31. Springer Series in Computational Mathematics. Springer, 2006.

[8] A. Chenciner and R. Montgomery. "A remarkable periodic solution of the three-body problem in the case of equal masses". In: *Annals of Mathematics* 152.3 (2000), pp. 881–901.

[9] R. A. Broucke. *Periodic orbits in the restricted three body problem with earth-moon masses*. Tech. rep. Nasa?, 1968.

[10] K. R. Meyer and D. C. Offin. *Introduction to Hamiltonian Dynamical Systems and the N-Body Problem*. Applied Mathematical Sciences. Springer Cham, 2017.

[11] Y. A. Kuznetsov. *Elements of applied bifurcation theory*. Springer, 1998.

[12] H. Goldstein, C. Poole, J. Safko, et al. *1., Classical Mechanics*. 1980.

[13] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.

[14] D. Boccaletti and G. Pucacco. *Theory of Orbits: Volume 1: Integrable Systems and Non-perturbative Methods*. Astronomy and Astrophysics Library. Springer Berlin Heidelberg, 1996.

[15] T. Kibble and F. H. Berkshire. *Classical mechanics*. world scientific publishing company, 2004.

[16] S. Greydanus, M. Dzamba, and J. Yosinski. *hamiltonian-nn*. `https://github.com/greydanus/hamiltonian-nn`. Accessed: 2026-01-18. 2019.

[17] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[18] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31 (2018).

[19] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[20] A. Zhu, P. Jin, and Y. Tang. "Deep Hamiltonian networks based on symplectic integrators". In: *arXiv preprint arXiv:2004.13830* (2020).

[21] R. A. Jacobs. "Increased rates of convergence through learning rate adaptation". In: *Neural networks* 1.4 (1988), pp. 295–307.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[23] A. Rahma, C. Datar, A. Cukarska, and F. Dietrich. "Rapid training of Hamiltonian graph networks without gradient descent". In: *arXiv preprint arXiv:2506.06558* (2025).