Algorithms of Hamiltonian neural networks

Victor Barbarich
Financial University
vktrbr@icloud.com

Abstract

Neural networks are integrating into our everyday life exceptionally quickly. We use voice assistant, self-driving car, computer vision, and much more every day. But, artificial intelligence has a little use in physics. The main reason is a regular error of the algorithms of ordinary neural network. Algorithms of Hamiltonian neural networks aimed at reducing errors and determine invariants. We will improve the third step in the scheme below.

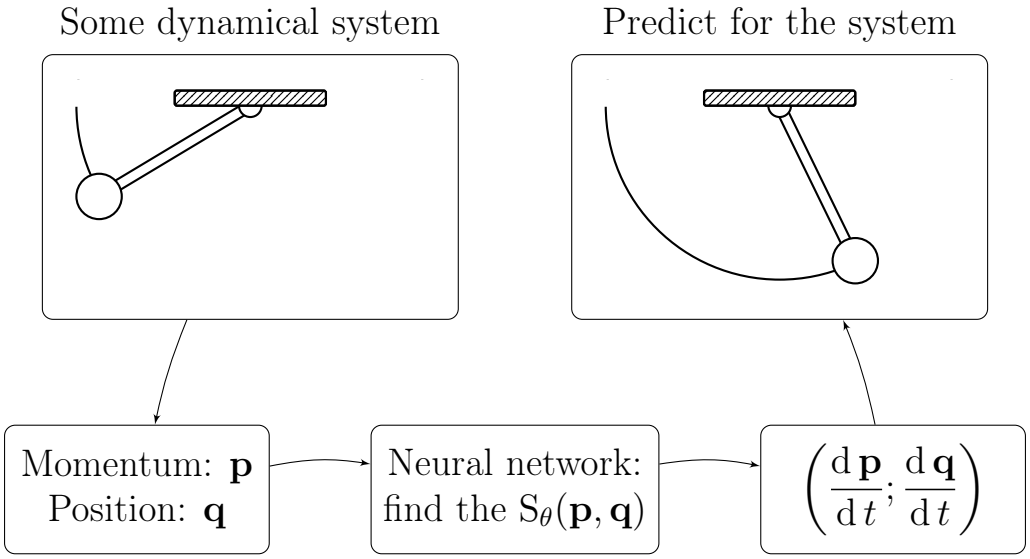

Figure 1: The scheme for predicting the state of a dynamical system.

Introduction

Hamiltonian

A dynamical system is a system that describes the states of objects in time and space. A Hamiltonian system is a dynamical system governed by Hamilton's equations.

The Hamiltonian $\mathcal{H}(\vec{p}, \vec{q}, t)$ is a function that depends on $2N+1$ variables (N is degrees of freedom), where $\vec{p} = (p_1, \ldots, p_N)$ is generalized momentum and $\vec{q} = (q_1, \ldots, q_N)$ is generalized position. But the Hamiltonian may not depend on time, then $\mathcal{H}(\vec{p}, \vec{q}, t) = \mathcal{H}(\vec{p}, \vec{q})$. Examples of such systems are the n-body problem or the pendulum.
So, we use the main property of the Hamiltonian in our neural network. This property related with the Hamilton's equations:

$$\dot{p}_j = \frac{dp_j}{dt} = -\frac{\partial \mathcal{H}}{\partial q_j}$$
$$\dot{q}_j = \frac{dq_j}{dt} = +\frac{\partial \mathcal{H}}{\partial p_j}$$

Then it naturally follows that if we know $(\vec{q}_0,\ \vec{p}_0)$ and $\left(\frac{d}{dt}\vec{q},\ \frac{d}{dt}\vec{p}\right)$, we can calculate $(\vec{q}_1,\ \vec{p}_1)$.

Neural network

In our work, we will use a neural network for found a function S_θ, that depends on a vector $x = (\vec{q}, \vec{p})$:

$$\mathrm{S}_\theta : \mathbb{R}^{2N} \longrightarrow \mathbb{R}^{2N}$$
$$\mathrm{S}_\theta : x = (\vec{q},\ \vec{p}\,) \longrightarrow \dot{x}_\theta$$
$$\dot{x}_\theta : \ \left\| \dot{x}_\theta - \left(\frac{d}{dt}\vec{q},\ \ \frac{d}{dt}\vec{p}\right) \right\|_2^2 < \epsilon, \quad \forall \epsilon > 0$$

Our objective is to found a S_θ. If we work with common architecture (let's named it by BNN), neural network will performs the roll of S_θ. This means that BNN takes a $(\vec{q}, \vec{p})$ and returns a $\dot{x}_\theta$.
Whereas Hamiltonian neural network (HNN) returns a Hamiltonian function of system. Let's denote HNN as H_θ. And then, we differentiate the H_θ by the $(\vec{q}, \vec{p})$ for find the $\dot{x}_\theta$.

Properties of the Hamiltonian that were used

Let, $x = [\, q_1 \;\; q_2 \;\; \cdots \;\; q_N \;\; p_1 \;\; p_2 \;\; \cdots \;\; p_N \,]^\top$, $\quad W = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix}$, where $\mathbf{0}$ is $N \times N$ zero matrix and $\mathbf{I}$ is $N \times N$ identity matrix.

In accordance with this and the property of the Hamiltonian, the time derivative of x:

$$\dot{x} = \begin{bmatrix} \frac{dq_1}{dt} \\ \vdots \\ \frac{dp_N}{dt} \end{bmatrix} = W \cdot \nabla_x \mathcal{H} = \begin{bmatrix} 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \\ -1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial q_1} \\ \vdots \\ \frac{\partial \mathcal{H}}{\partial p_N} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial p_1} \\ \vdots \\ -\frac{\partial \mathcal{H}}{\partial q_N} \end{bmatrix}$$

Let we know x_0 (the initial state) and $\frac{dx}{dt} = \dot{x}$ (the evolution), then:

$$x_1 = x_0 + \int_{t_0}^{t_1} \dot{x} dt = x_0 + x \Big|_{t_0}^{t_1} = x_0 + x_1 - x_0 = x_1$$

For conservative systems (their phase space does not shrink over time), the Hamilton function represents the total energy, that is, the sum of the kinetic and potential energies of the system. $\mathcal{H} = T + U$.
Next, we will test HNN on two system.

The pendulum

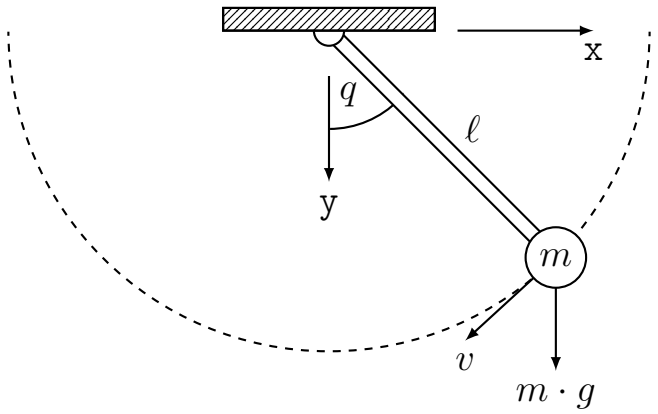

Figure 2: The pendulum. q is the angle of deviation from the ordinate.

This is the simplest Hamiltonian system. A good example to demonstrate the work of HNN.

Let's describe the system:

$$U = m \cdot g \cdot l \cdot (1 - \cos q); \qquad T = \frac{m \cdot v^2}{2} = \frac{m \cdot l^2 \cdot \dot{q}^2}{2}$$

$$p = \frac{\partial L}{\partial \dot{q}} = \frac{\partial (T - U)}{\partial \dot{q}} = m \cdot l^2 \cdot \dot{q}; \qquad \dot{q} = \frac{p}{m \cdot l^2}$$

$$\mathcal{H} = T + U = \frac{p^2}{2m \cdot l^2} + m \cdot g \cdot l \cdot (1 - \cos q)$$

$$\dot{p} = -\frac{\partial \mathcal{H}}{\partial q} = -m \cdot g \cdot l \cdot \sin q$$

Let the mass of the pendulum be 1 kg ($m = 1$), and the length is 1 m ($l = 1$). The Hamiltonian of the pendulum system is:

$$\mathcal{H} = \frac{p^2}{2} + g(1 - \cos q); \qquad \dot{x} = \begin{bmatrix} p \\ -g \cdot \sin q \end{bmatrix}$$

The double pendulum

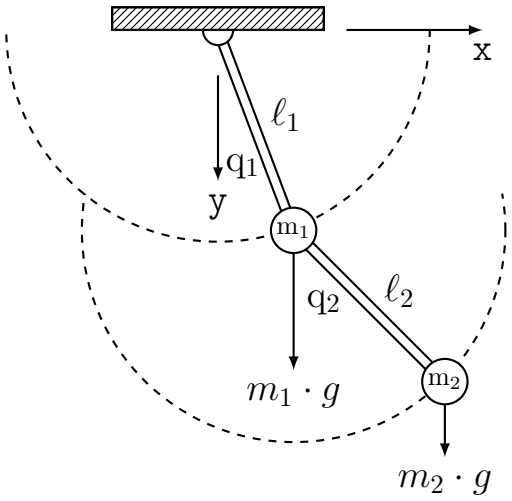

Figure 3: The double pendulum. q_1, q_2 are the angles of deviation from the ordinate.

A double pendulum without dissipation is a Hamiltonian system that is chaotic. This means that the smallest change in the initial state will give an unpredictable system.

The Hamiltonian of the double pendulum system:

$$\mathcal{H} = \frac{p_1^2 m_2 l_2^2 - 2p_1 p_2 m_2 l_1 l_2 \cos(q_1 - q_2) + p_2^2 (m_1 + m_2) l_1^2}{2m_2 l_1^2 l_2^2 \left[m_1 + m_2 \sin^2(q_1 - q_2)\right]} - \\ - (m_1 + m_2) g l_1 \cos q_1 - m_2 g l_2 \cos q_2.$$

Let the mass of the pendulums be 1 kg ($m_1 = m_2 = 1$), and the length are 1 m ($l_1 = l_2 = 1$).

$$\dot{x} = \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial p_1} & \frac{\partial \mathcal{H}}{\partial p_2} & -\frac{\partial \mathcal{H}}{\partial q_1} & -\frac{\partial \mathcal{H}}{\partial q_2} \end{bmatrix}^\top$$

Hamiltonian neural network

Firstly, let's say about «Universal approximation theorem», which was proved by George Cybenko.

Cybenko's Theorem.

1. Let I_n denote the n-dimensional unit cube, $I_n = [0, 1]^n$.
   The space of measurable functions on I_n is denoted $L^1(I_n)$.
2. Let σ is sigmoidal if: $\sigma(x) \longrightarrow \begin{cases} 1 & \text{as } x \longrightarrow +\infty \\ 0 & \text{as } x \longrightarrow -\infty \end{cases}$
3. $\alpha_j, \theta_j \in \mathbb{R}$; $y_j \in \mathbb{R}^n$ is some coefficient.

Theorem: Let σ be bounded measurable sigmoidal function. Then finite sums of the form:

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^\top x + \theta_j)$$

are dense in $L^1(I_n)$. In other words, given any $f \in L^1(I_n)$ and $\epsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$\|G - f\|_{L^1} = \int_{I_n} |G(x) - f(x)| dx < \epsilon$$

This is a extremely important theorem for neural networks. In a sense, $G(x)$ is a neural network.

Matrix form of a neural network

1. Let the neural network first layer, called $a^{(1)}$, be a column vector of $m \times 1$ dimension and second layer, called $a^{(2)}$, be a column vector of $k \times 1$ dimension.

2. $\mathrm{Y} = \left(y_{ij}\right) \in \mathbb{R}^{k \times m}$ is a weight matrix. $\mathrm{b} = \left(b_k\right) \in \mathbb{R}^{k \times 1}$ is a bias matrix.

3. Let $\sigma\left(\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}^\top\right) = \begin{bmatrix} \sigma(a_1) & \sigma(a_2) & \cdots & \sigma(a_n) \end{bmatrix}^\top$

Then the second layer of the neural network is equal to:

$$a^{(2)} = \sigma\left(Y \cdot a^{(1)} + b\right)$$

In practice, we often use neural networks from several layers. We will use a three-layer neural network when working with a simple pendulum (the first layer is the input layer, the second layer is the hidden layer, and the third layer is the output layer). When working with a double pendulum we will use a four-layer neural network. This solution will allow us to find function faster, and this property was discovered empirically.

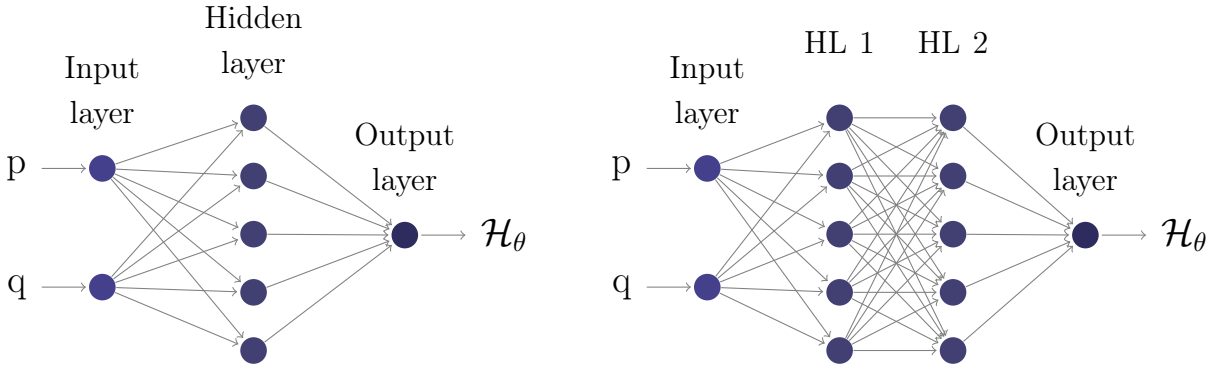

Figure 4: Left: HNN architecture for a single pendulum.
Right: HNN architecture for a double pendulum.

HNN for a pendulum in matrix form: $H_\theta = \begin{bmatrix} \alpha_1 & \cdots & \alpha_k \end{bmatrix} \cdot \sigma(Y \cdot x + b) =$

$$= \begin{bmatrix} \alpha_1 & \cdots & \alpha_k \end{bmatrix}_{1\times k} \cdot \sigma \underbrace{\left(\begin{bmatrix} y_{ij} \end{bmatrix}_{k\times 2} \cdot \underbrace{\begin{bmatrix} q \\ p \end{bmatrix}_{2\times 1}}_{\text{input layer}} + \begin{bmatrix} b_j \end{bmatrix}_{k\times 1} \right)}_{\text{hidden layer}}$$

And by virtue of Universal approximation theorem, such y_{ij}, α_j, b_j exist.

Determining the weights

The main problem of a work with neural network is to determine the weight matrices, bias columns and coefficient columns. We use "supervised learning". This is the method that maps an input to an output. Algorithm:

1. *Data generation.* We create many pairs of $x = (q, p)^\top$ and $\dot{x} = (\dot{q}, \dot{p})^\top$ (the number of pairs is denoted by T). Let's denote the set of $x^\top$ as $X \in \mathbb{R}^{T \times 2N}$ and set of $\dot{x}^\top$ as $\dot{X} \in \mathbb{R}^{T \times 2N}$. Instead of creating data, you can perform many measurements of a dynamic system.

2. We calculate $W \cdot \nabla_x \mathcal{H}_\theta \quad$ for all $x \in X$. We denote this as $\dot{X}_\theta$.

3. Consider the function: $\text{MSE}(Y, b, \alpha) = \|\dot{X} - \dot{X}_\theta\|_F^2 = \sum_{i=1}^{T} \|\dot{x}_i - \dot{x}_{i,\theta}\|_2^2$.

4. After, we minimize the MSE by weight matrices, bias columns and coefficient columns, using, for example, gradient methods or others. $\text{MSE} \longrightarrow \min_{Y,b,\alpha}$.

Finally, when we have chosen the correct dimensions for the weight matrices, and have trained the neural network acceptably well, we can use it to predict the evolution in dynamical systems.

Results

Single pendulum

Hamiltonian neural network settings:

- Hidden layer dimension is 700. 10 000 epochs.
- $\sigma(x) = \frac{1}{1+e^{-x}}$ is standard logistic function.
- The Adaptive Moment Estimation (ADAM) minimize MSE.

Hamiltonian neural network results:

- Last train MSE ≈ 0.008. The total training time is 12 minutes.
- Test MSE is equal ≈ 0.0078. (50 different random initial states)

Graphs

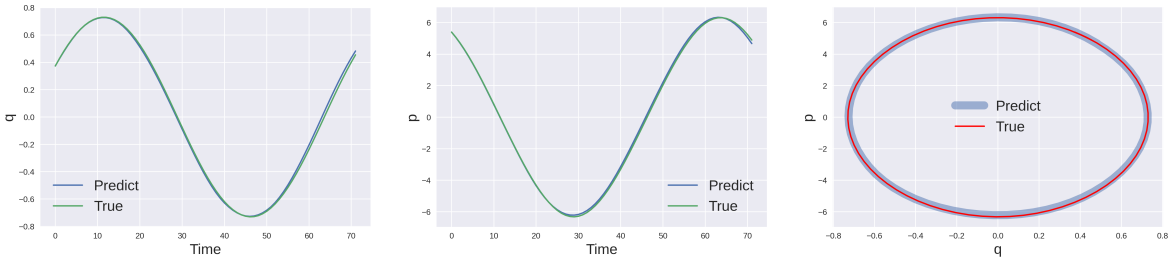

Figure 5: Prediction based on random test data

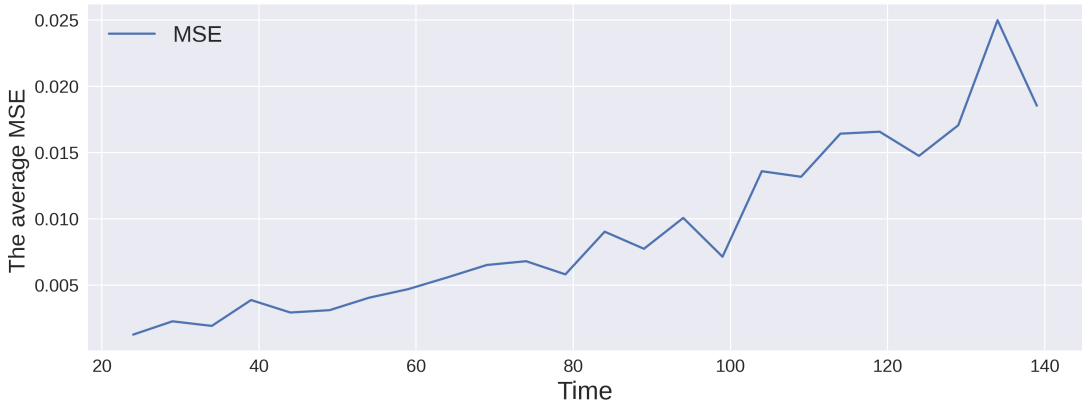

Figure 6: The MSE prediction is based on 50 random test data and with different prediction duration.

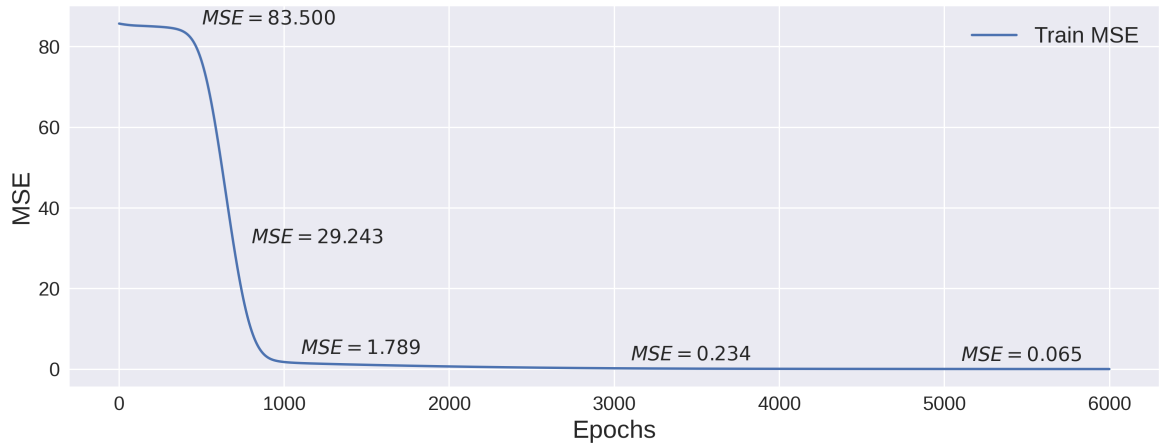

Figure 7: MSE training on different epochs.

Double pendulum

Hamiltonian neural network settings:

- Hidden layers: 1 - 3000, 2 - 400. 8 000 epochs.
- $\sigma(x) = \dfrac{1}{1+e^{-x}}$.
- The Adaptive Moment Estimation (ADAM) minimize MSE.

Hamiltonian neural network results:

- Last train MSE ≈ 0.01 . The total training time is 11 hours.

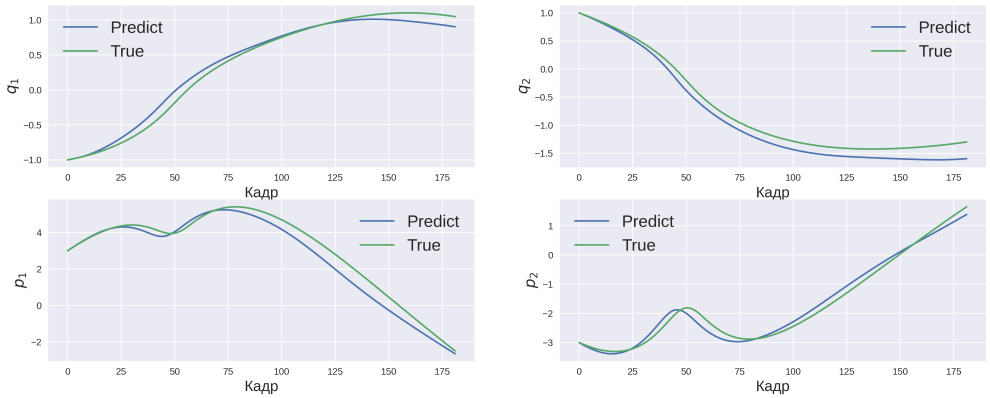

Figure 8: Prediction based on random test data

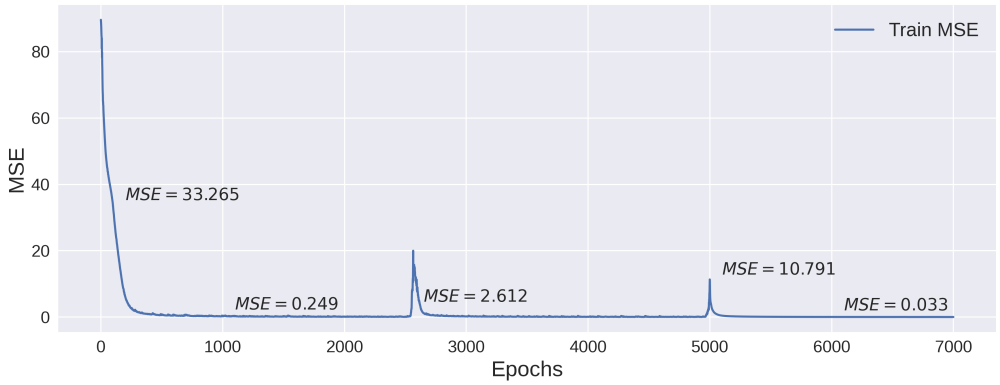

Figure 9: MSE training on different epochs.

Conclusion

The Hamiltonian neural network is a powerful basic tool for studying dynamical systems without dissipation. HNN can help in the study of planetary systems, gases and other dynamics. We can consider HNN as an additional tool for testing theories. But this requires additional research.

References

[1] *Sam Greydanus, Misko Dzamba, Jason Yosinski*, Hamiltonian Neural Networks, *arXiv.org*, 2019. https://arxiv.org/abs/1906.01563

[2] *Peter Toth, Danilo J. Rezende, Andrew Jaegle, Sebastien Racaniere, Aleksandar Botev, Irina Higgins*, HAMILTONIAN GENERATIVE NETWORKS, *Conference paper at ICLR 2020*

[3] *Cybenko, G*, "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals, and Systems.* 1989 link

[4] *J.R.Dormand, P.J.Prince*, A family of embedded Runge-Kutta formulae *arXiv*, 1980 https://www.sciencedirect.com/science/article/pii/0771050X80900133?