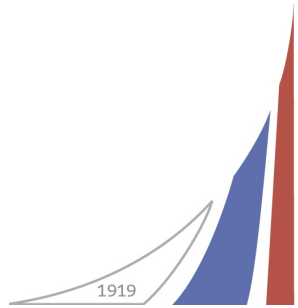


FINANCIAL UNIVERSITY

UNDER THE GOVERNMENT OF THE RUSSIAN FEDERATION

«Algorithms of Hamiltonian neural networks»

Barbarich Victor

Topological methods in dynamics and related topics

August 3, 2021

Single pendulum

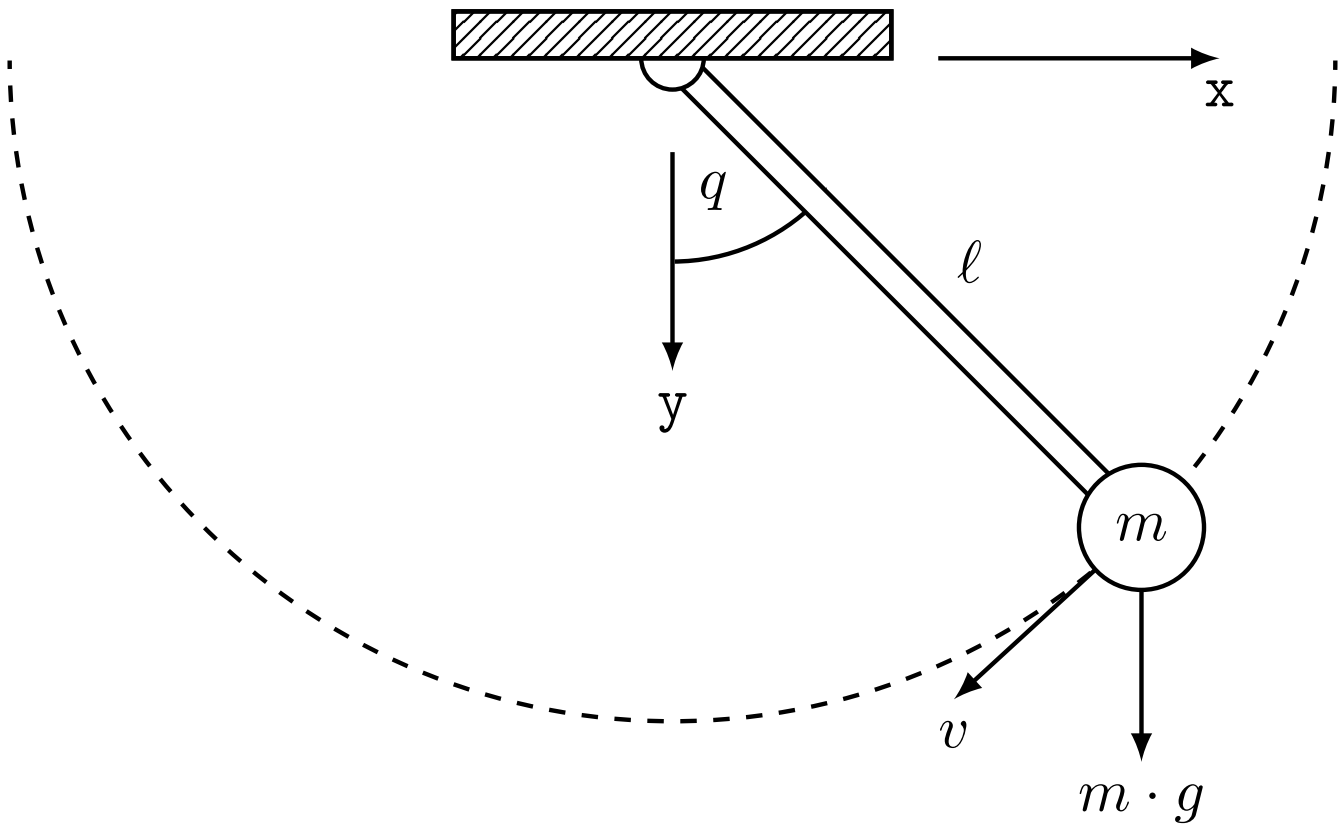


Рис.: The pendulum. q is the angle of deviation from the ordinate.

Hamiltonian of a SP

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

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

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

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

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

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

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

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

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

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

Hamiltonian of a SP

$$\mathcal{H}(q, p, t) = \mathcal{H}(q, p)$$

The main property that we use in our work:

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

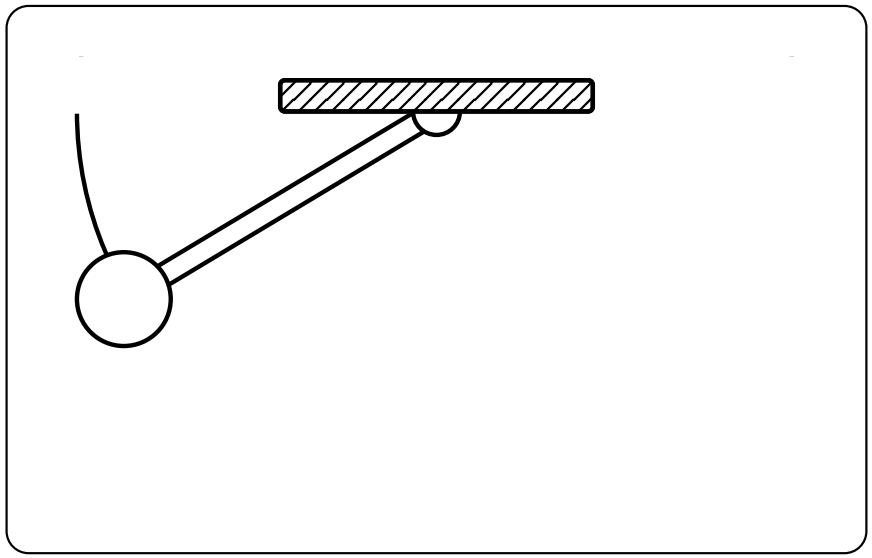
Let:

$$x = [\, q_1 \quad q_2 \quad \cdots \quad q_N \quad p_1 \quad p_2 \quad \cdots \quad p_N \,]^\top, \quad W = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix},$$

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

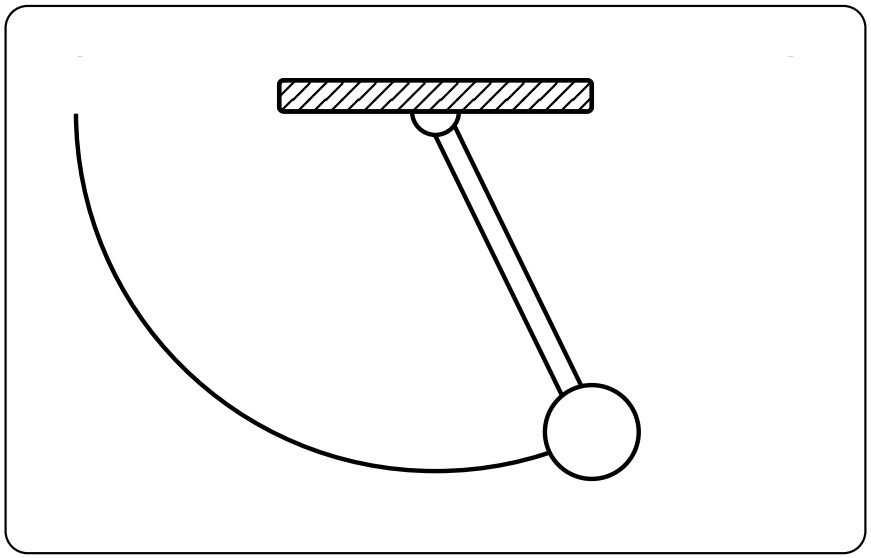
$$\dot{x} = \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial p_1} \\ \vdots \\ -\frac{\partial \mathcal{H}}{\partial q_N} \end{bmatrix}$$

Some dynamical system

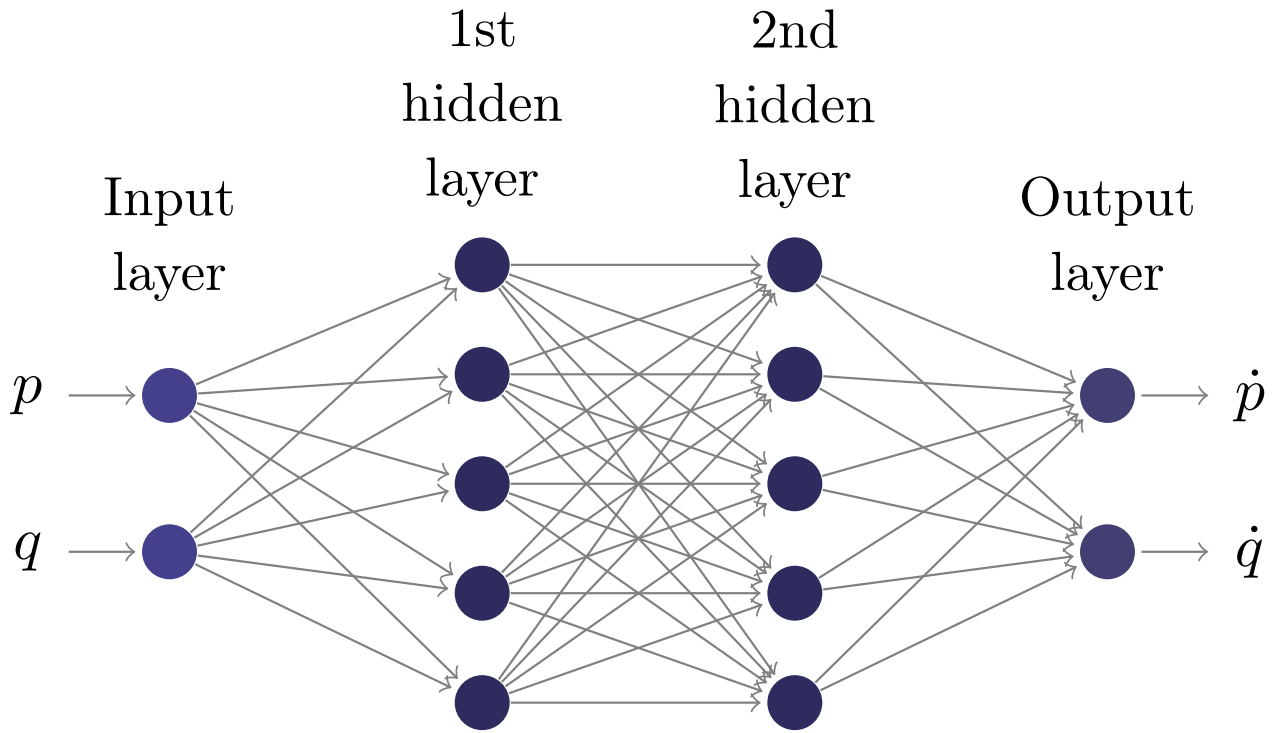


Some kind of algorithm

Predict for the system



We can consider a neural network as a function
$S_\theta(q,p): \quad \mathbb{R}^{2N} \longrightarrow \mathbb{R}^{2N}$.



Baseline neural network

So, the baseline neural network is a function, that depends of $x_0 = [q_0\ p_0]^\top$, and return a $\dot{x}_0 = [\dot{q}_0\ \dot{p}_0]^\top$

So, not strictly:

$$\mathrm{S}_\theta : \mathbb{R}^{2N} \longrightarrow \mathbb{R}^{2N}$$

$$\mathrm{S}_\theta : x = (\vec{q}, \vec{p}) \longrightarrow \dot{x}_\theta$$

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

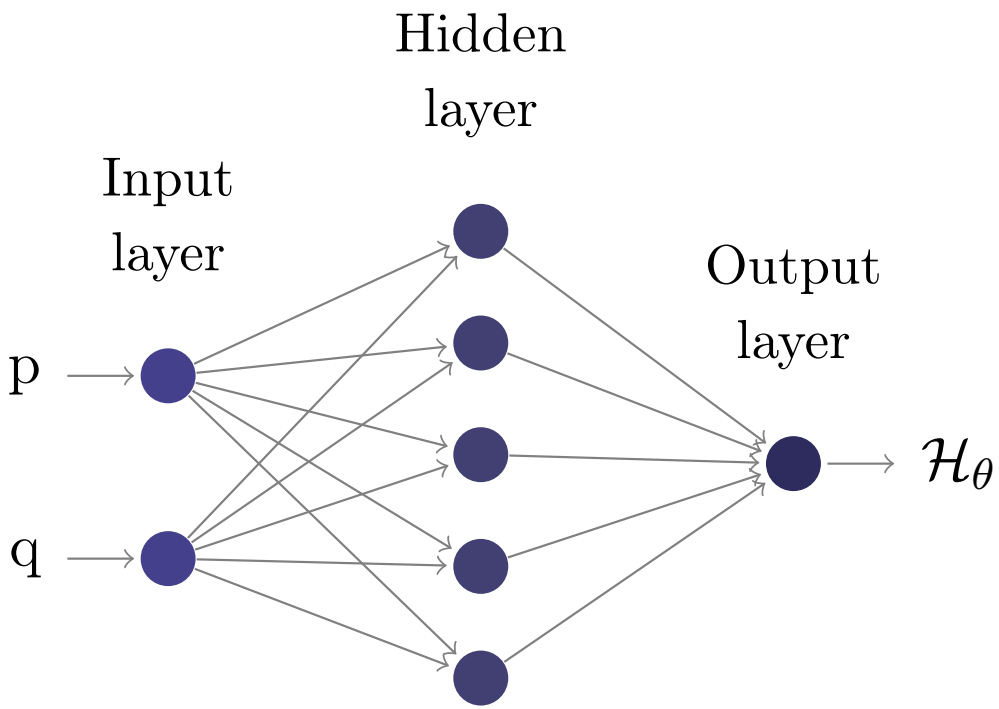
Disadvantages:

1. We can't find the earlier states of the system
2. The neural network doesn't save the total energy of the system.
3. Practical application has shown that the predictions are really bad based on data that was not in the training.

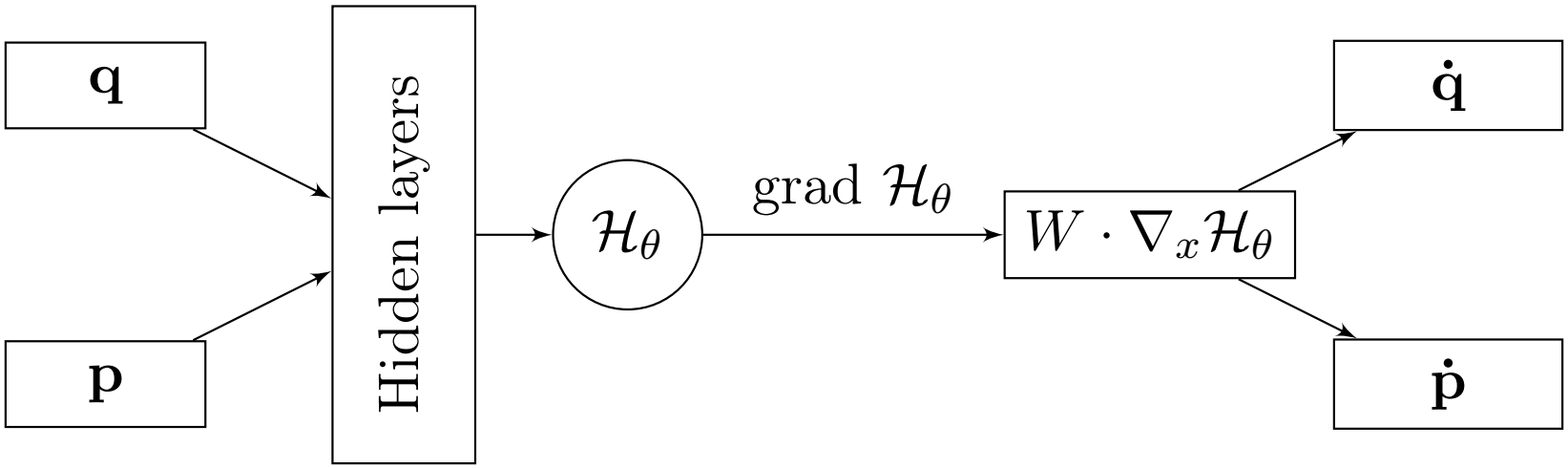
Advantages:

1. Simple code implementation.

HNN can solve these problems.
Let's denote this neural network as $\mathcal{H}_\theta$



HNN architecture for a single pendulum.



Scheme of a Hamiltonian neural network

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

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

Matrix form of a neural network

Let:

$a^{(1)} = \left[a_{ij}\right] \in \mathbb{R}^{m \times 1}, \quad a^{(2)} = \left[a_{ij}\right] \in \mathbb{R}^{k \times 1}$ (1st and 2nd layers.)

Matrix form of a neural network

Let:

$a^{(1)} = \left[a_{ij}\right] \in \mathbb{R}^{m\times 1}, \quad a^{(2)} = \left[a_{ij}\right] \in \mathbb{R}^{k\times 1}$ (1st and 2nd layers.)

$Y = \left[y_{ij}\right] \in \mathbb{R}^{k\times m}, \quad b = \left[b_k\right] \in \mathbb{R}^{k\times 1}$ (weights and biases).

Let:

$a^{(1)} = \left[a_{ij}\right] \in \mathbb{R}^{m\times 1}, \quad a^{(2)} = \left[a_{ij}\right] \in \mathbb{R}^{k\times 1}$ (1st and 2nd layers.)

$Y = \left[y_{ij}\right] \in \mathbb{R}^{k\times m}, \quad b = \left[b_k\right] \in \mathbb{R}^{k\times 1}$ (weights and biases).

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

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

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

HNN for a pendulum in matrix form:

$$H_\theta = \begin{bmatrix} \alpha_1 & \cdots & \alpha_k \end{bmatrix} \cdot \sigma(Y \cdot x + b) =$$

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

HNN for a pendulum in matrix form:

$$H_\theta = \begin{bmatrix} \alpha_1 & \cdots & \alpha_k \end{bmatrix} \cdot \sigma(Y \cdot x + b) =$$

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

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

How can we determine weights and biases?

1. *Data generation.* We create T pairs of $x = (q, p)^\top$ and $\dot{x} = (\dot{q}, \dot{p})^\top$ Let's denote the set of all $x^\top$ as $X \in \mathbb{R}^{T \times 2N}$ and set of all $\dot{x}^\top$ as $\dot{X} \in \mathbb{R}^{T \times 2N}$.

1. *Data generation.* We create T pairs of $x = (q, p)^\top$ and $\dot{x} = (\dot{q}, \dot{p})^\top$ Let's denote the set of all $x^\top$ as $X \in \mathbb{R}^{T \times 2N}$ and set of all $\dot{x}^\top$ as $\dot{X} \in \mathbb{R}^{T \times 2N}$.

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

1. *Data generation.* We create T pairs of $x = (q, p)^\top$ and $\dot{x} = (\dot{q}, \dot{p})^\top$ Let's denote the set of all $x^\top$ as $X \in \mathbb{R}^{T \times 2N}$ and set of all $\dot{x}^\top$ as $\dot{X} \in \mathbb{R}^{T \times 2N}$.

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

3. Consider the function:

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

1. *Data generation.* We create T pairs of $x = (q, p)^\top$ and $\dot{x} = (\dot{q}, \dot{p})^\top$ Let's denote the set of all $x^\top$ as $X \in \mathbb{R}^{T \times 2N}$ and set of all $\dot{x}^\top$ as $\dot{X} \in \mathbb{R}^{T \times 2N}$.

2. We calculate $W \cdot \nabla_x \mathcal{H}_\theta$ 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.

$$\text{MSE} \longrightarrow \min_{Y,b,\alpha}$$

1. *Data generation.* We create T pairs of $x = (q, p)^\top$ and $\dot{x} = (\dot{q}, \dot{p})^\top$ Let's denote the set of all $x^\top$ as $X \in \mathbb{R}^{T \times 2N}$ and set of all $\dot{x}^\top$ as $\dot{X} \in \mathbb{R}^{T \times 2N}$.

2. We calculate $W \cdot \nabla_x \mathcal{H}_\theta$ 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.

$$\text{MSE} \longrightarrow \min_{Y,b,\alpha}$$

5. Repeat 2-4 steps for the required epsilon.

Link to the bot. You can look at a neural network with your initial data. Scan the QR code.

HNN results on a single pendulum

Hamiltonian neural network settings:

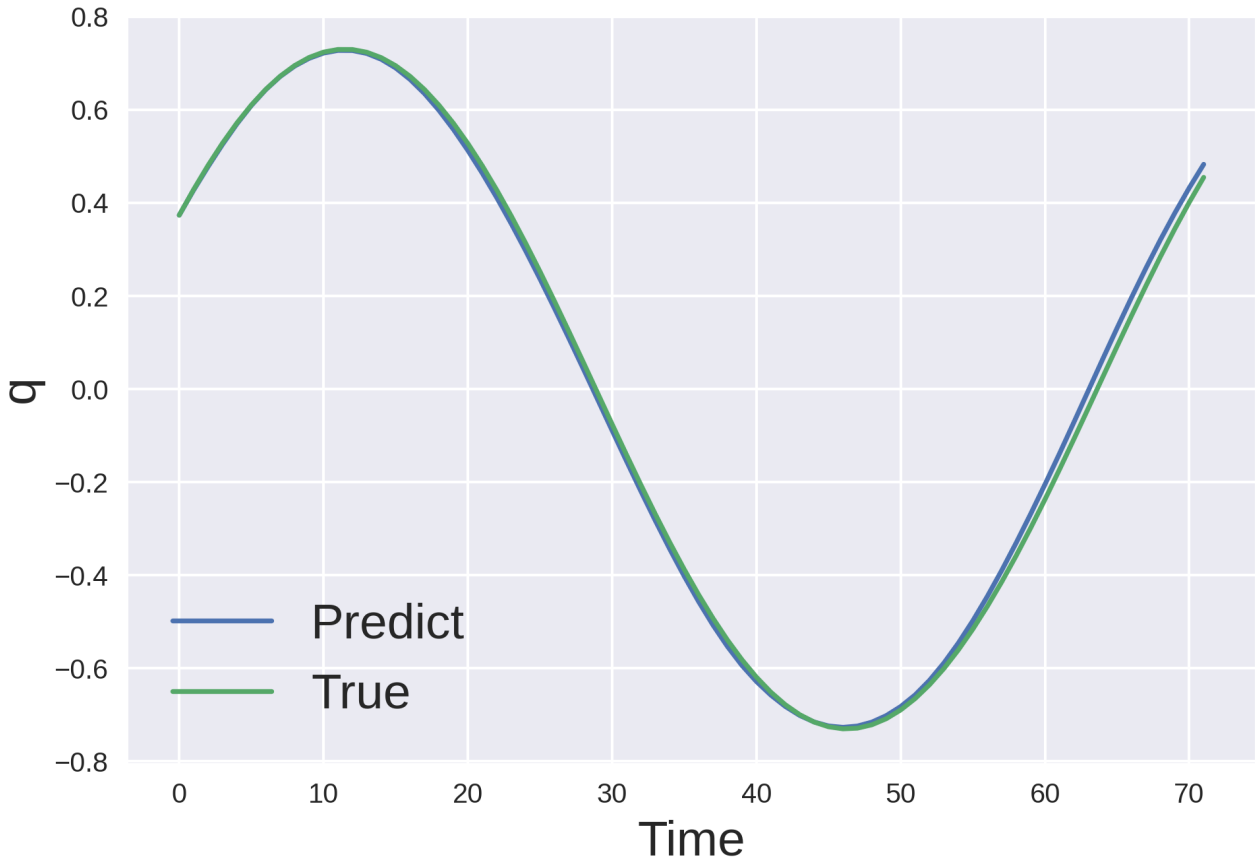
1. Hidden layer dimension is 700. 10 000 epochs.
2. $\sigma(x) = \frac{1}{1+e^{-x}}$ is standard logistic function.
3. The Adaptive Moment Estimation (ADAM) minimize MSE.

HNN results on a single pendulum

Hamiltonian neural network results:

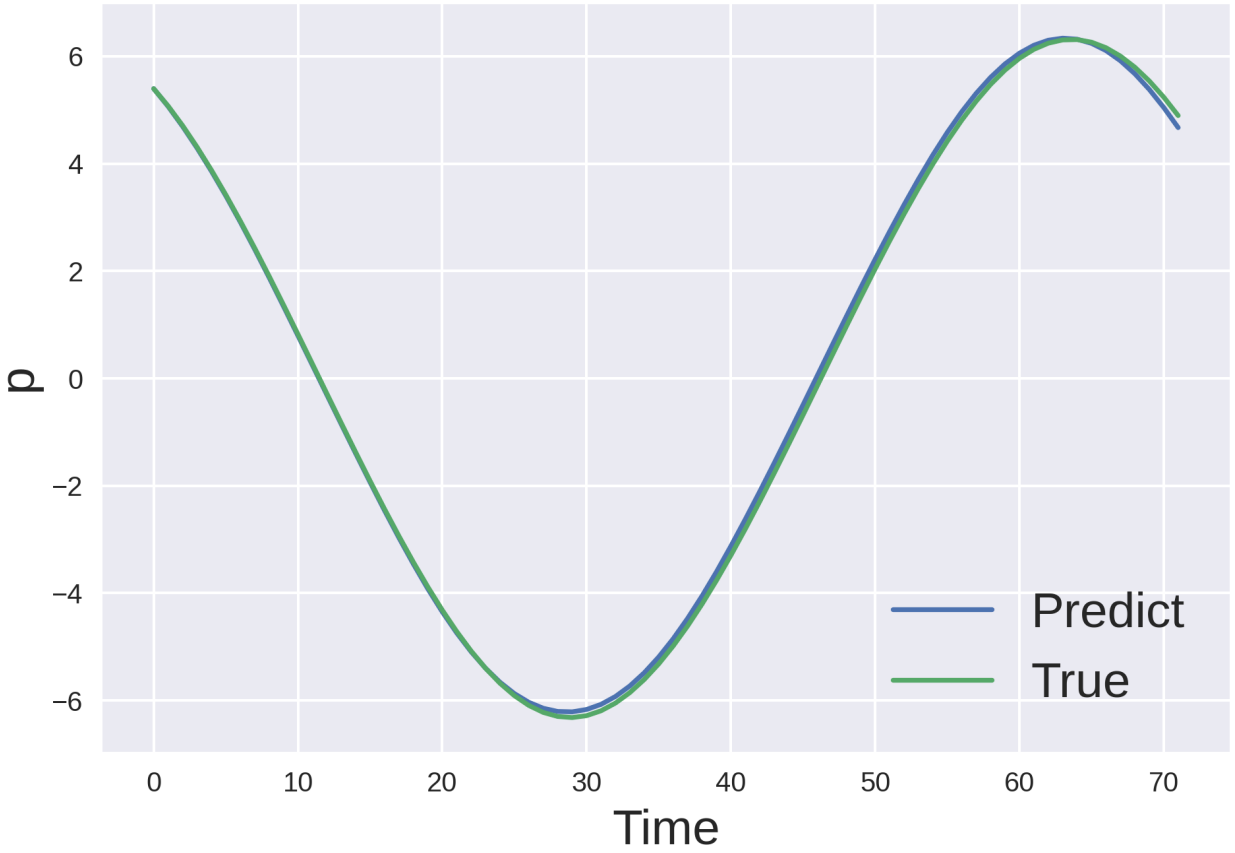
1. Last train MSE ≈ 0.008. The total training time is 12 minutes.
2. Test MSE is equal ≈ 0.0078. (50 different random initial states)

HNN results on a single pendulum



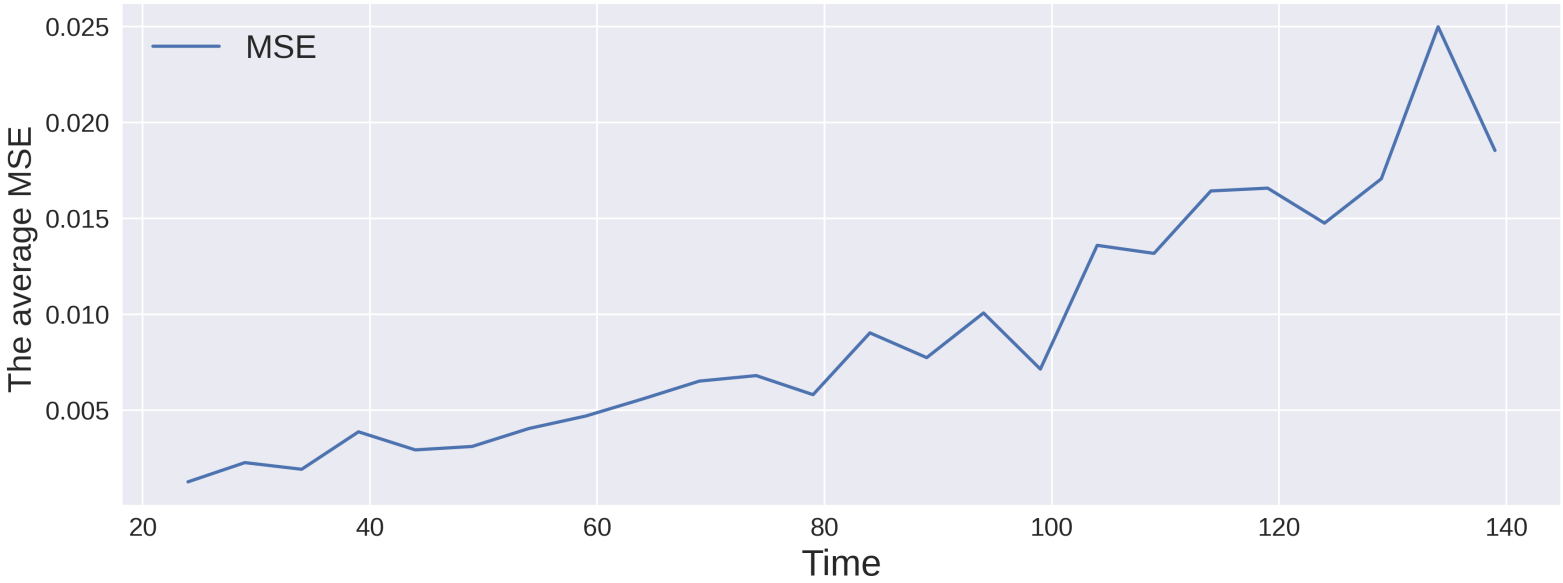
Prediction based on random test data

HNN results on a single pendulum



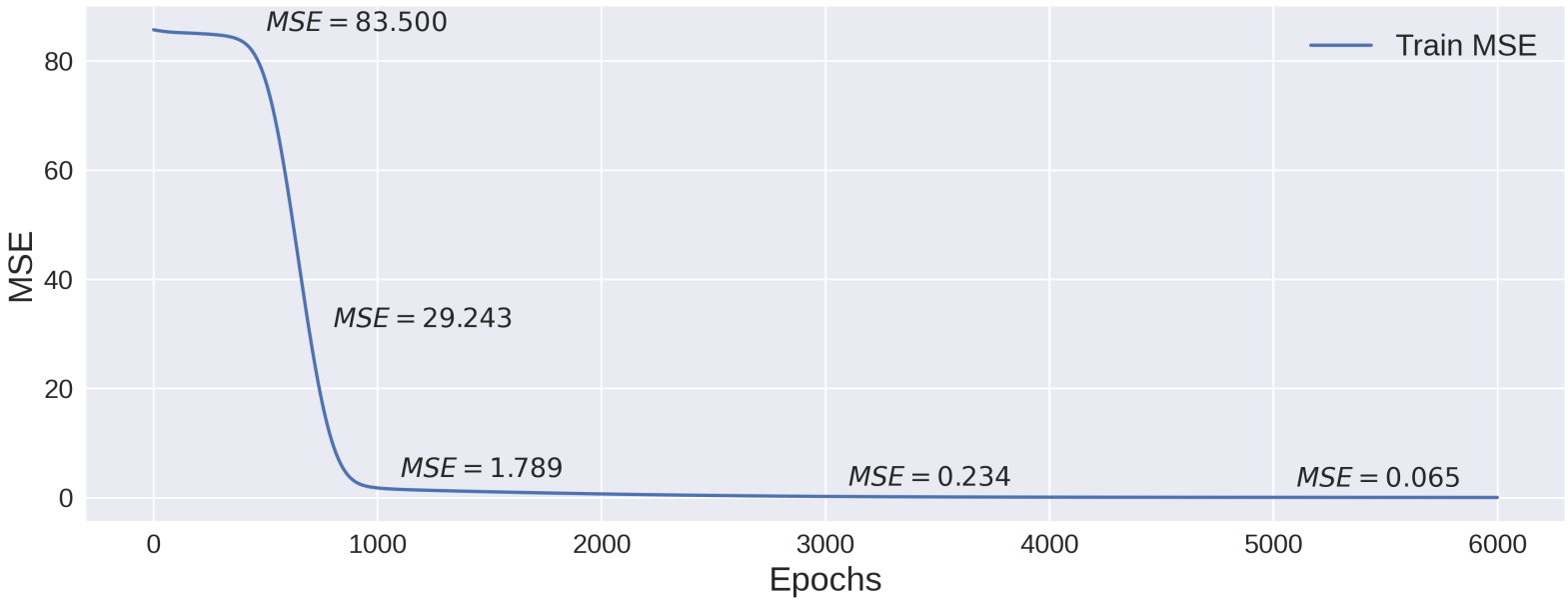
Prediction based on random test data

HNN results on a single pendulum



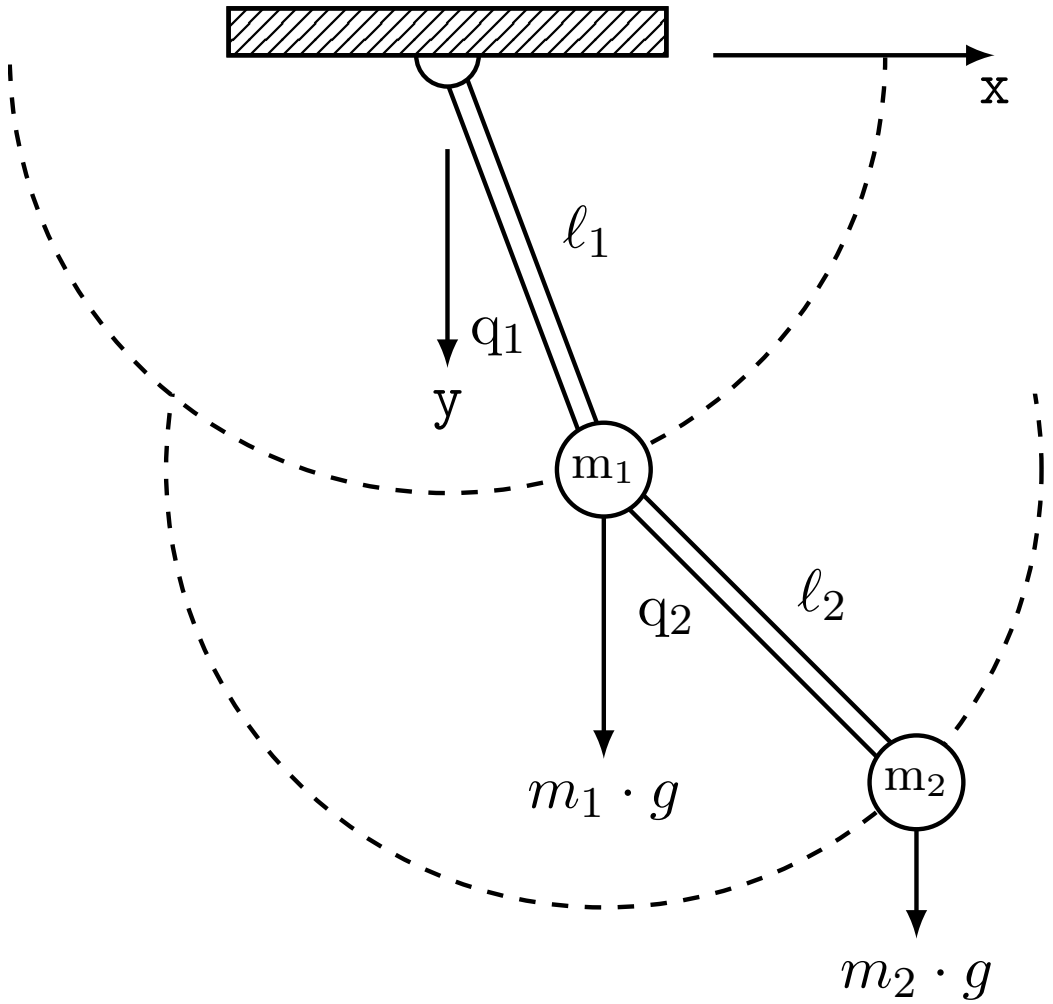
The MSE prediction is based on 50 random test data and with different prediction duration.

HNN results on a single pendulum

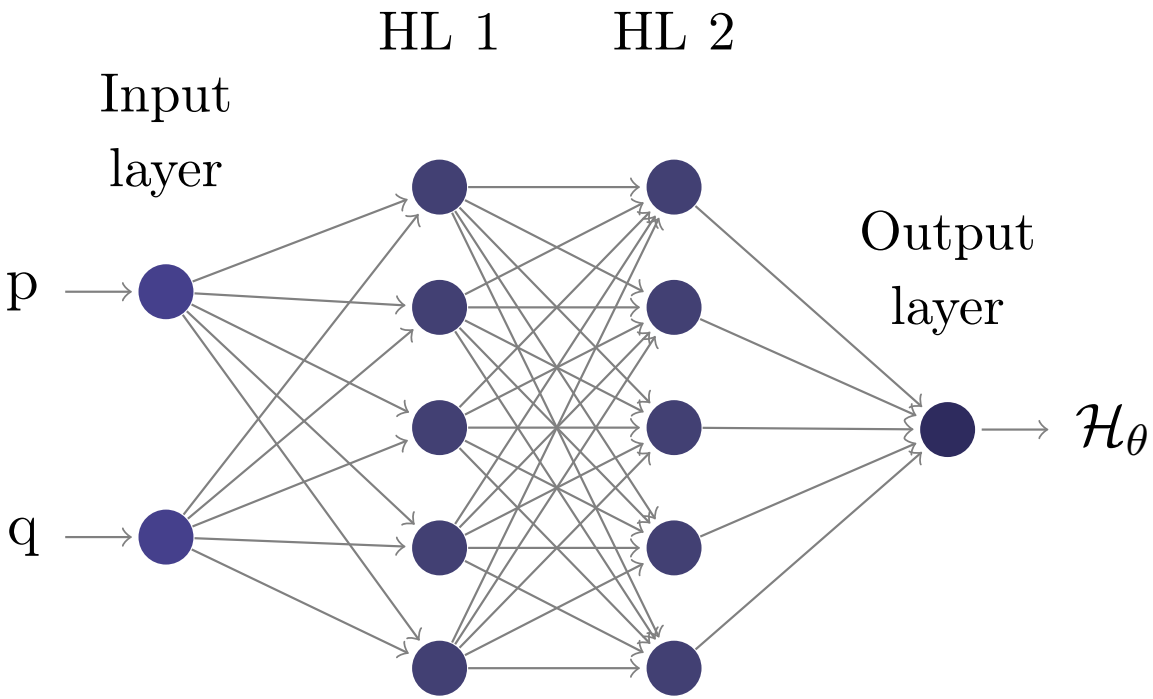


MSE training on different epochs.

HNN results on a double pendulum



HNN results on a double pendulum



HNN architecture for a double pendulum.

HNN results on a double pendulum

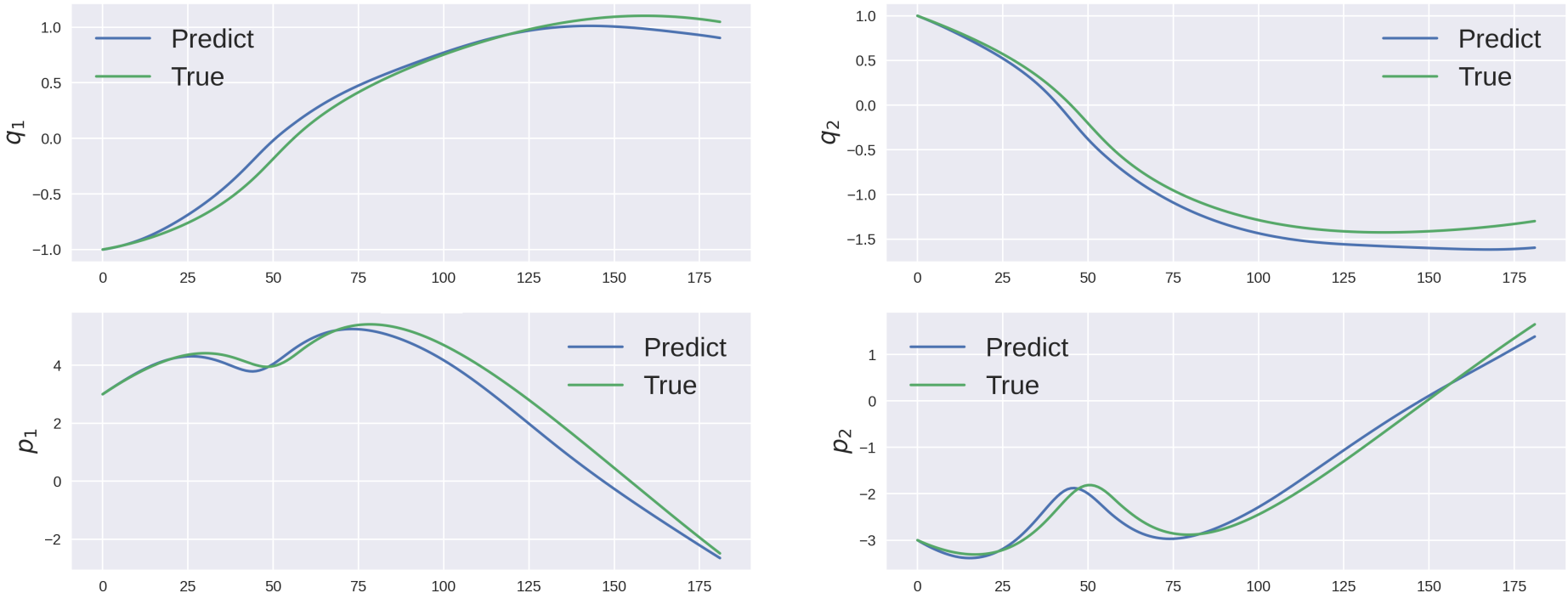
Hamiltonian neural network settings:

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

Hamiltonian neural network results:

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

HNN results on a double pendulum



Prediction based on random test data

Thanks for attention!

Moscow 2021