Hamiltonian Neural Networks: Modeling Pendulum Dynamics

Yuri Paglierani

University of Trieste

September, 04, 2024

Outline

Introduction to Hamiltonian Neural Networks

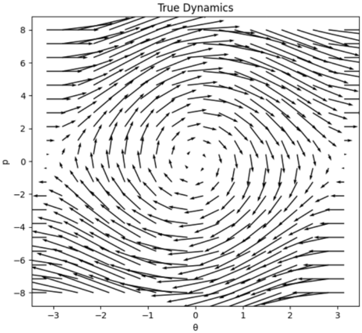
- Neural networks that build an Hamiltonian representation, and then use it for making predictions;
- They are a possible path for exploiting symmetries, and constraints in Neural Nets;
- Key advantages:
  - Long-term stability in predictions
  - Physically consistent results
  - Generalizable to various Hamiltonian systems
- Main topic of today: Application to single and double pendulum systems

Hamiltonian Mechanics: A Brief Overview

- Hamiltonian function:
  $H(q, p) = T(q, p) + V(q)$
  - $T(q, p)$: Kinetic energy (sometimes doesn't depend on q)
  - $V(q)$: Potential energy
- Hamilton's equations:

$$\frac{dq}{dt} = \frac{\partial H}{\partial p}, \quad \frac{dp}{dt} = -\frac{\partial H}{\partial q}$$

- Energy conservation: $\frac{\mathrm{d}H}{\mathrm{d}t} = 0$ for isolated systems



Hamiltonian Neural Networks (HNNs)

- Learn the Hamiltonian function from data
- Network architecture enforces Hamilton's equations
- Training process:
  - Input: State variables (q, p)
  - Output: Predicted Phase space $(\dot{q}, \dot{p})$
  - Loss: $\mathcal{L} = \left\| \frac{\partial \hat{H}}{\partial p} - \dot{q} \right\|^2 + \left\| \frac{\partial \hat{H}}{\partial q} + \dot{p} \right\|^2$

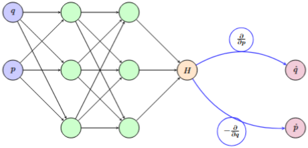


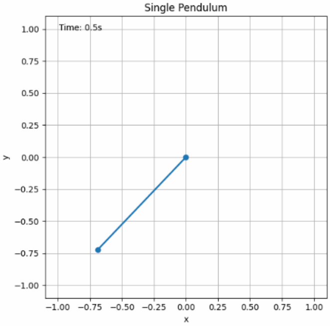
Figure 1: Hamiltonian Neural Network with Inputs, Hidden Layers, and Outputs

Single Pendulum: Problem Setup

- Simple Pendulum
- State variables:
  - $q = \theta$ (angle)
  - $p = ml^2\dot{\theta}$ (angular momentum)
- Hamiltonian:

$$H = \frac{p^2}{2ml^2} + mgl(1 - \cos\theta)$$

- Phase space:
  - $\dot{q} = \frac{p}{ml^2}$
  - $\dot{p} = -mgl\sin\theta$

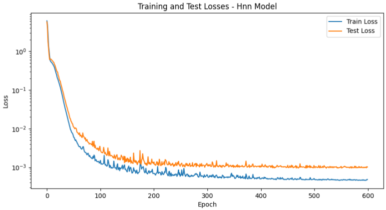


Single pendulum setup

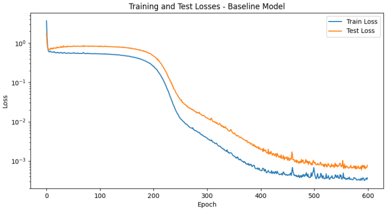
Single Pendulum: HNN Implementation

- Data generation:
  - Done in Jax, 150 trajectories, 16 steps each, Stormer-Verlet integrator
  - Store state variables, phase space predictions, and energies
  - Add a bit gaussian noise to each trajectory
- HNN and MLP (Baseline) architecture:
  - Input layer: 2 features (q, p)
  - Hidden layers: 3 layers with 200 neurons each, softplus activation
  - Output layer: 2 targets (time derivatives of the features)
- Training:
  - Optimizer: Adam
  - Scheduler: Exponential decay ($\gamma = 0.9$, patience=10)
  - Learning rate: 1e-3
  - Batch size: 128
  - Epochs: 600

Single Pendulum: Training Landscape



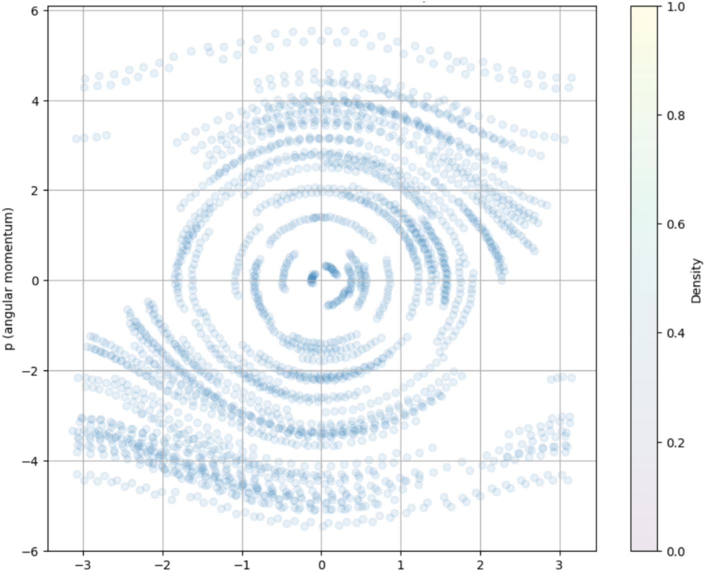
HNN



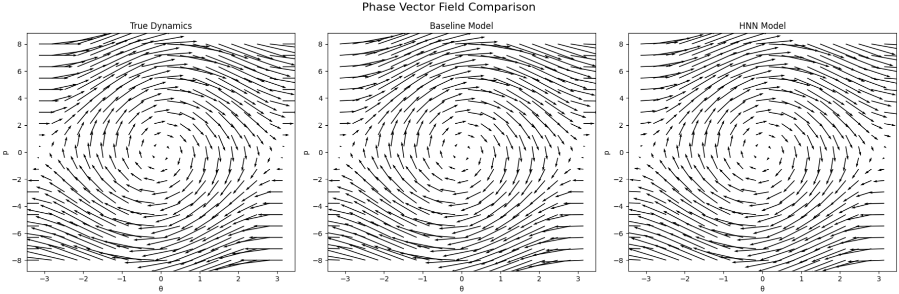
MLP

- HNN reaches a plateau faster than the MLP;
- They both reach the same order of magnitude of Loss on plateau;
- We are overfitting a bit;

Single Pendulum: True Sample Space

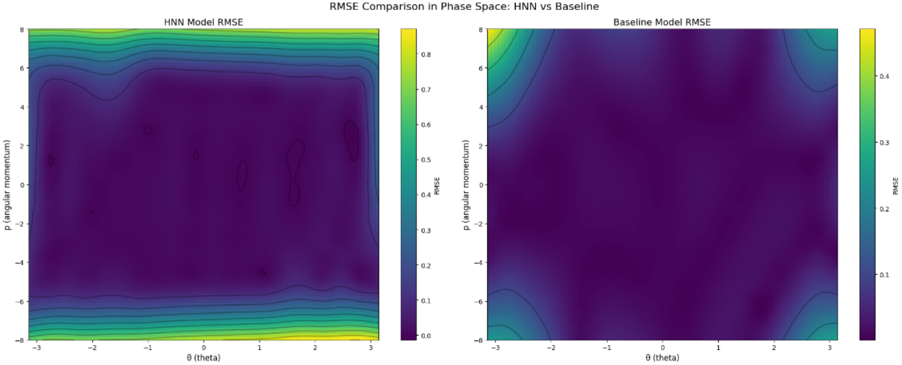


Single Pendulum: Phase space comparison

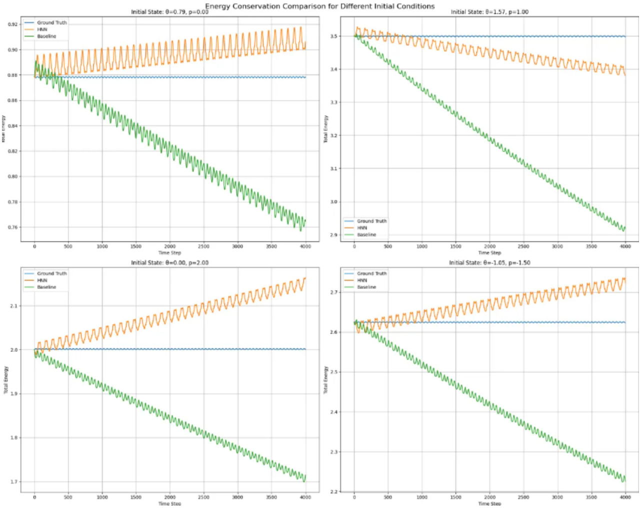


Also the phase space look similar.

Single Pendulum: RMSE comparison

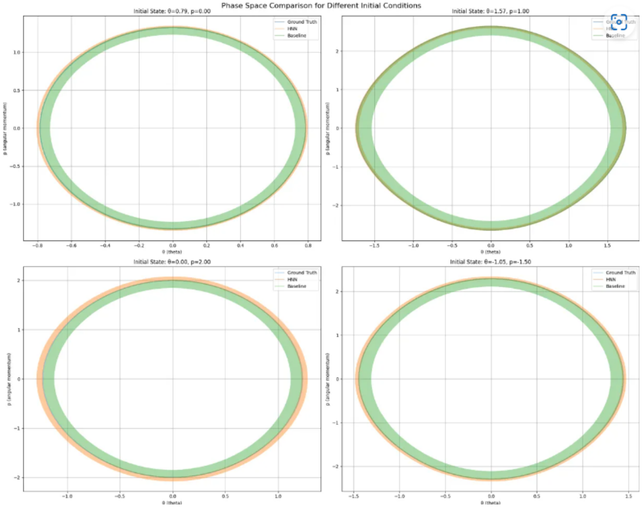


Single Pendulum: Long Term Energy comparison



The baseline dissipates during time!

Single Pendulum: Long Term Phase space trajectory

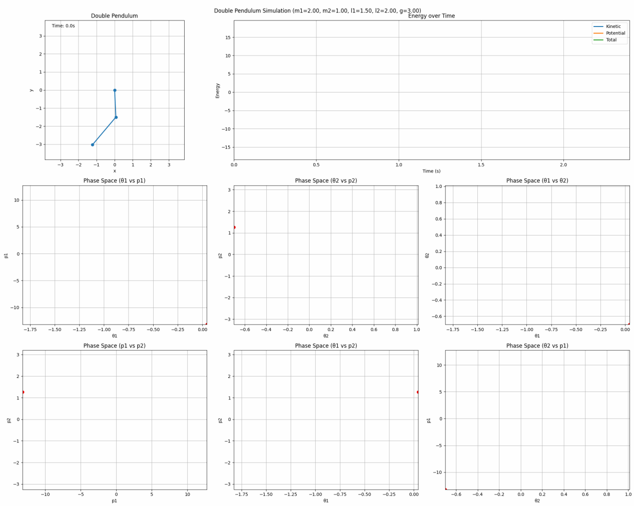


Double Pendulum: Problem Setup

- Chaotic system with two coupled pendulums
- State variables:
  - q_1, q_2 (angles)
  - p_1, p_2 (angular momenta)
- Way more complex Hamiltonian function:

$$H = \frac{m_2 l_2^2 p_{\theta_1}^2 + (m_1 + m_2) l_1^2 p_{\theta_2}^2 - 2 m_2 l_1 l_2 p_{\theta_1} p_{\theta_2} \cos(\theta_1 - \theta_2)}{2 m_2 l_1^2 l_2^2 [m_1 + m_2 \sin^2(\theta_1 - \theta_2)]} \\ - (m_1 + m_2) g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2 \tag{1}$$

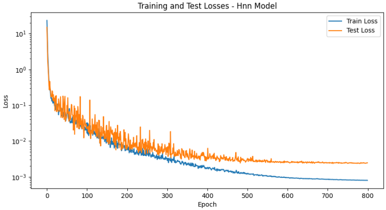
Double Pendulum Setup

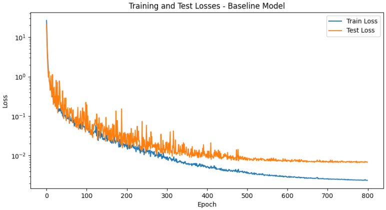


Double Pendulum: HNN Implementation

- Data generation:
  - Done in Jax, 900 trajectories, 61 steps each, Stormer-Verlet integrator
  - Store state variables, phase space predictions, and energies
- HNN and MLP (Baseline) architecture:
  - Input layer: 4 features (q_1, q_2, p_1, p_2)
  - Hidden layers: 4 layers with 200 neurons each, softplus activation
  - Output layer: 4 targets (time derivatives of the features)
- Training:
  - Optimizer: Adam
  - Scheduler: Exponential decay ($\gamma = 0.9$, patience=10)
  - Learning rate: 1e-3
  - Batch size: 128
  - Epochs: 800

Double Pendulum: Training Landscape

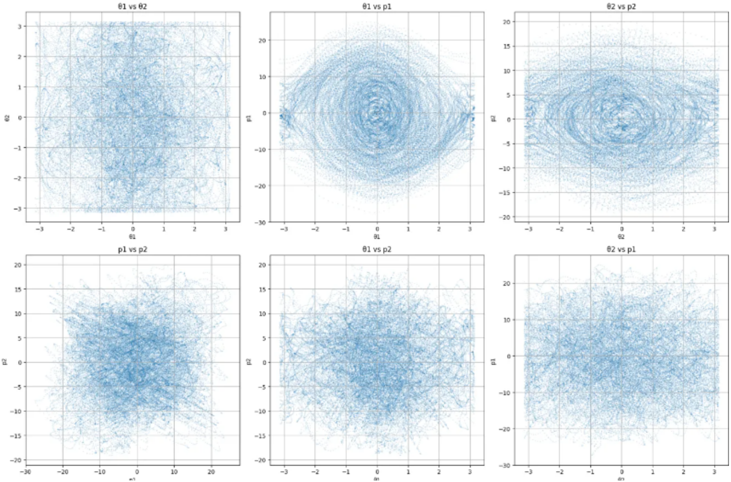




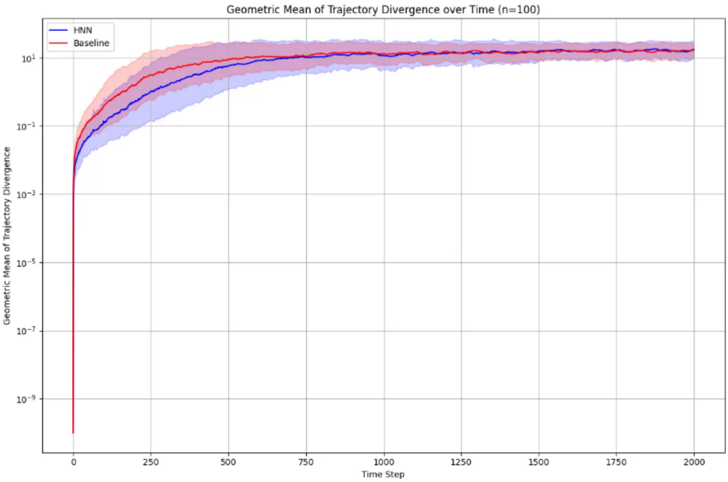
HNN MLP

- Here you can see that the HNN dominates the Baseline (by 1 order of magnitude);
- We are overfitting a bit;

Double Pendulum: True Sample Space



Double Pendulum: Divergence Plot in time



HNN vs. Traditional Neural Networks

- Comparison metrics:
  - Mean Squared Error (MSE)
  - Energy conservation behavior (for single pendulum)
  - Long-term stability of predictions
  - Time for each training iteration
  - Time for each inference iteration
  - Divergence plot across simulation (double pendulum)

HNN vs. Traditional Neural Networks

- Results:
  - HNN in general shows lower MSE if compared with the Baseline
  - In single pendulum we notice that the Baseline dissipates in time, instead the HNN moves on a approximately closed trajectory in sample space (single pendulum)
  - Due to the computation of partial derivatives, the HNN is in general slower (approximately by a factor of 2 if compared with the baseline)
  - For the same reason, also the inference time of the HNN is worse than the baseline (3 times slower)
  - However, as you can see from the loss landscapes the HNN trains faster (approximately 2x speed wrt the Baseline), and reaches better minimum
  - You can see from the divergence plot of the double pendulum that even if the HNN is slower, it's more time coherent wrt the Ground Truth, at least for the first 700 trajectories, then the 2 dynamics show the same divergence

Challenges and Future Work

- Challenges:
  - Handling more complex physical systems
  - Improving computational efficiency for real-time applications
- Future work:
  - Extend to other Hamiltonian systems (e.g., N-body problems)
  - Incorporate uncertainty quantification
  - Explore applications in control systems and robotics
  - Explore how it behave the HNN with different architectures

Conclusion

- HNNs effectively model pendulum dynamics:
  - Accurate predictions for both single and double pendulums
  - Are more physically plausible
  - Outperform traditional neural networks in Hamiltonian systems
- Key advantages:
  - Physics-informed machine learning
  - Long-term stability in predictions
  - Generalizable to various Hamiltonian systems
- Promising approach for simulating and understanding complex physical systems

Thank You!

Questions?

For more information:
https://github.com/YuriPaglierani/Hamiltonian-Neural-Networks-Pendulums