EE5111 Project

May 21, 2025

1 Introduction

In this project, we aim to model the time evolution of physical systems starting from a given set of initial conditions. Our primary focus in this project is to leverage **Hamiltonian mechanics** for modeling dynamical systems through the conservation of energy. To implement this idea, we use two main approaches:

- Linear models
- Neural Networks (specifically concept of Hamiltonian Neural Networks (HNNs))

Among these, neural networks are particularly well-suited due to their ability to approximate complex functions and generalize from data. These models are inspired by the structure of Hamiltonian mechanics and are designed to learn dynamics that inherently respect conservation laws—especially the conservation of energy—in an unsupervised manner.

2 Linear Models

We use LASSO regression for its **sparsity-inducing** property, enabling interpretable models with a small number of significant terms, while reducing overfitting. We model the Hamiltonian $\hat{H}(q,p)$ as a polynomial:

$$\hat{H}(q,p) = \sum_{i=0}^{n}\sum_{j=0}^{n-i} \alpha_{ij}\, q^i p^j = \Phi(q,p)^T W,$$

where $\Phi(q,p)$ are basis terms and W is the coefficient vector.

Using Hamilton's equations:

$$\dot{q}_t = \frac{\partial \hat{H}}{\partial p}, \quad \dot{p}_t = -\frac{\partial \hat{H}}{\partial q},$$

we define:

$$y_1 = X_1 W, \quad y_2 = X_2 W,$$

with X_1, X_2 as feature matrices obtained from derivatives of $\hat{H}$.

The optimization problem becomes:

$$\min_W \left\| \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} W - \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\|_2^2 + \lambda \|W\|_1.$$

3 Neural Networks

The **Baseline Neural Network** directly maps the state variables (q_t, p_t) to their time derivatives $(\dot{q}_t, \dot{p}_t)$ using supervised learning. While this approach is simple and effective in some scenarios, it does not incorporate any physical structure or constraints. As a result, the model can produce predictions that violate fundamental physical laws, such as **conservation of energy**.

In contrast, the **Hamiltonian Neural Network (HNN)** learns the underlying Hamiltonian function $H(q, p)$ of the system. The time derivatives are then computed (using a simple chain in the neural network) to get Hamilton's equations:

$$\dot{q} = \frac{\partial H}{\partial p}, \quad \dot{p} = -\frac{\partial H}{\partial q}.$$

Apparently, in need to conserve energy, the model underperformed as far as reducing trajectory loss is concerned, so we came up with the following strategies:

3.1 Improvements over Standard HNN

To enhance the performance and generalizability of the Hamiltonian Neural Network, we introduced following improvement:

- Various regularizing techniques (like Dropout, add a penalty term $\lambda \|W\|_2$ to the loss function, Early Stopping) to prevent overfitting / over-parameterizing the model.
- **Trajectory-Based Losses:** In addition to the standard HNN loss based on Hamilton's equations, we incorporated trajectory supervision to guide learning:
  1. **Full-Trajectory Loss:** The model predicts the entire trajectory from initial conditions, and the loss accumulates across all time steps.
  2. **One-Step Prediction Loss:** At each time step, the model predicts only the next state based on the current true state, and loss is computed step-by-step.
  3. k^{th}**-Step Prediction Loss:** At each time step, the model predicts the next k steps based on the current true state, and loss is computed step-by-step. (we didn't implement it but we thought it to be a more generalised idea)

  Including this loss function kind of creates a balance in between conserving energy and predicting the trajectory path.

- **Final Loss Function:** All components were combined into a total loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{HNN}} + \alpha\,\mathcal{L}_{\text{traj}} + \lambda\,\|W\|_2^2,$$

where $\mathcal{L}_{\text{traj}}$ is either the full-trajectory or one-step loss, α controls the trajectory loss weight, and λ controls weight decay.

3.2 Modelling Dissipative Dynamics

To model dissipative dynamic systems we decided to model Damped Pendulum. In reality, a pendulum is subject to dissipative forces like friction and air resistance, which leads to a gradual loss in its total mechanical energy. To account for energy dissipation in Hamiltonian Neural Networks (HNNs), we introduced an additional frictional force term. The damped pendulum dynamics are governed by the equation:

$$\frac{d^2\theta}{dt^2} + \lambda\frac{d\theta}{dt} + \omega^2\sin\theta = 0 \quad \Rightarrow \quad \frac{dp}{dt} = -\frac{\partial H}{\partial q} + F_{\text{friction}}(p, q)$$

Here, the frictional force is modeled using a separate neural network, referred to as the Friction Network. This network learns to capture non-conservative forces present in the system.

The modified HNN loss function incorporating friction becomes:

$$\mathcal{L}_{\text{HNN+Fric}} = \left\| \frac{\partial H}{\partial q_t} + F_{\text{friction}}(p_t, q_t) + \dot{p}_t \right\|_2^2 + \left\| \frac{\partial H}{\partial p_t} - \dot{q}_t \right\|_2^2$$

This formulation allows the model to learn both conservative (Hamiltonian-based) and non-conservative (friction-based) components of the dynamics.

4 Langragian Neural Network (LNN's)

While HNNs are powerful tools for modeling conservative dynamical systems, they face challenges when applied to systems with higher degrees of freedom like the double pendulum:

- Deriving and differentiating the Hamiltonian for higher-order systems can lead to cumbersome symbolic expressions and unstable learning.
- For second-order systems like the double pendulum, Hamiltonian-based training is not straightforward due to the indirect nature of generalized coordinates.

LNNs offer the following advantages for complex systems:

- They operate directly in position-velocity space $(q, \dot{q})$, avoiding the need to switch to momentum space.

- They naturally handle second-order dynamics, which are more intuitive for many physical systems.
- They are better suited for modeling systems with constraints and coupled nonlinear dynamics, such as the double pendulum.

Limitations of Lagrangian Neural Networks (LNNs)

- **Expensive Hessian computation:** Calculating $\ddot{q}$ from the Lagrangian requires computing the Hessian, which is computationally intensive and increases inference time.
- **Costly numerical integration:** Since LNNs predict second-order dynamics, integrating to the next time step involves more complex numerical solvers, further adding to runtime.
- **Performance difference over time:** LNN tends perform better than HNN only in longer periods of time. For estimating the dynamics for short period of time, LNN does not seems to be a viable solution given its performance over head. .

5 Code

Our codebase can be accessed from `https://github.com/me21b172/ET_Project.git`. Please refer to the *hamiltonian_nn* folder and follow Readme.