Learning Dynamics using Hamiltonian Inspired Neural Networks

Emma Hayes[1], Mathias Heider [2], Carrie Vanty[3],

[1]Carnegie Mellon University, [2] University of Delaware, [3] Middlebury College

EMORY

Motivation

- We wish to further extend Ruthotto *et al*'s Hamiltonian Inspired Neural Networks (HINN) [3] to predict the value of the Hamiltonian
- We build this framework using the fundamental relationships of a Hamiltonian Ordinary Differential Equation:

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

- The Verlet method can be used with these relationships to gather information about the dynamics of our problems [1]
- Learning the Hamiltonian would give more insight into the dynamics of the given data set, allowing us to see how well our Neural Network conserves a quantity analogous to total energy

Learning Problem

- Our neural network approximates the Hamiltonian

$$[t, p_t, q_t] \xrightarrow{\text{ResNet}} \mathcal{H}_\theta$$

  Where θ are the network parameters we wish to optimize
- We use the autograd feature from the hessQuik package to calculate the partials [4]
- These partials are used for our Verlet discretization in forward propagation:

$$p_{\theta+1} = p_\theta + h\frac{\partial \mathcal{H}_\theta}{\partial q}, \qquad q_{\theta+1} = q_\theta - h\frac{\partial \mathcal{H}_\theta}{\partial p}$$

- We can use the p_θ, q_θ to learn the parameters θ by minimizing:

$$\min_\theta \int_0^T \left[|p_\theta - p_{true}|^2 + |q_\theta - q_{true}|^2 \right] dt$$

Experiments

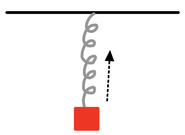

Fig. 1: Simple Spring-Mass System

To test our neural network, we first began with the ideal spring mass system, a Hamiltonian system which simulates the movement of a mass attached to a spring

The two body problem is another Hamiltonian system that predicts the motion between two masses, where the only the forces acting on the masses are the gravitational forces from the other mass

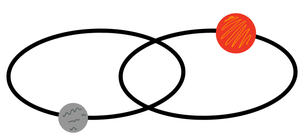

Fig. 2: Two Body Problem

Results

To do the time stepping, we take input t_0, p_{t_0}, q_{t_0} and use the neural network with optimal parameters θ^* to get $\mathcal{H}_{\theta^*}$. Then use the Verlet method to find the p and q values at the next time step. Repeat this process until we reach our final time step, giving us the entire p and q.

Spring Mass System

- In our approach we use a residual neural network structure created using the hessQuik [4] and PyTorch [5] packages. The residual neural network has a depth of 20 layers and width of 3
- Sample Size: 2000 || Train Size: 2000 || Test Size: 300 || Validation Size: 300
- Epoch: 1500 || Optimizer: Gradient Descent || Learning Rate: $1 * 10^{-3}$

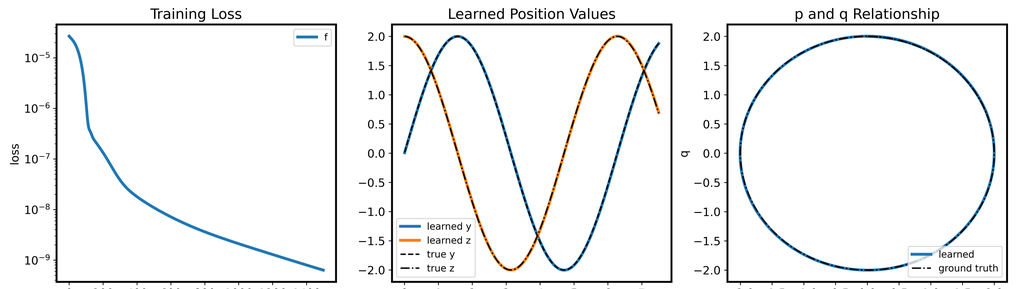

Fig. 3: Spring Mass System

Two-Body Problem

- We then approached the two-body problem with our Hamiltonian Inspired Neural Network. We used the same residual neural network structure with 20 layers and a width of 3
- Sample Size: 4000 || Train Size: 4000 || Test Size: 2900 || Validation Size: 900
- Epoch: 1500 || Optimizer: Gradient Descent || Learning Rate: $1 * 10^{-3}$

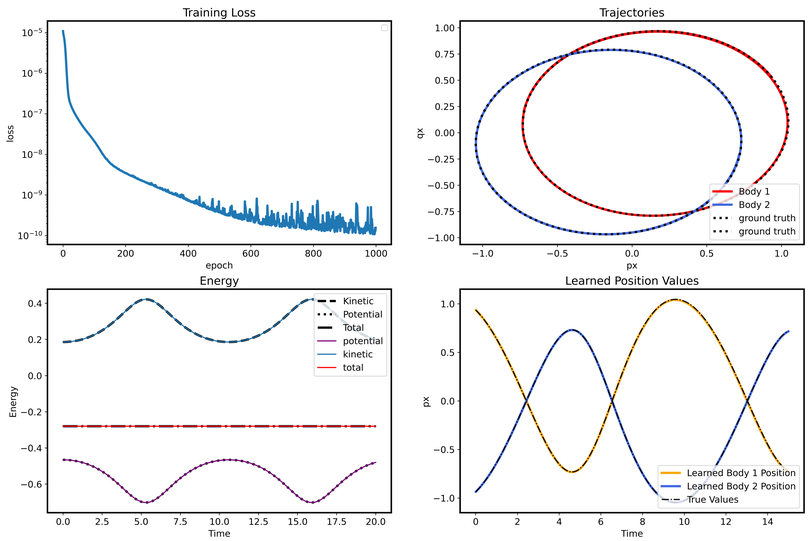

Fig. 4: Two Body Problem

Comparison

In order to evaluate how our method performs, we want to compare to a similar method by Greydanus *et al* that also found the value of a Hamiltonian for a given data set [2]. However, our method differs greatly from theirs in that they:

- Implement L_2 loss for training and test losses. $\mathcal{H}_\theta$ is again the predicted Hamiltonian from the network

$$\mathcal{L}_{HNN} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial p} - \frac{\partial q}{\partial t} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial q} + \frac{\partial p}{\partial t} \right\|_2$$

- Then they use the RK4 method to predict position and velocity

Conclusion

- We extended the HINN to approximate the value of the Hamiltonian
- The Verlet method was implemented to discretize the ODE that results from a Hamiltonian system
- We used our network to learn the Hamiltonian for a spring mass system and the two-body problem, and found that our system learns the dynamics of the given Hamiltonian data
- Moving forward, we hope to further apply our method on more complicated Hamiltonian systems, for example the three body problem

Acknowledgements

This work is supported in part by the US National Science Foundation award DMS-2051019. We would like to thank our mentor Deepanshu Verma and all other mentors for their guidance and support. We would also like to thank the other 2022 REU/RET participants for their company, conversation, and constant encouragement.

References

[1] Uri M. Ascher. *Numerical methods for evolutionary differential equations*. Vol. 5. Computational Science & Engineering. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008, pp. xiv+395. ISBN: 978-0-898716-52-8. DOI: 10.1137/1.9780898718911. URL: https://doi.org/10.1137/1.9780898718911.

[2] Sam Greydanus, Misko Dzamba, and Jason Yosinski. "Hamiltonian Neural Networks". In: *ArXiv* abs/1906.01563 (2019).

[3] Eldad Haber and Lars Ruthotto. "Stable architectures for deep neural networks". In: *Inverse Problems* 34.1 (2018), pp. 014004, 22. ISSN: 0266-5611. DOI: 10.1088/1361-6420/aa9a90. URL: https://doi.org/10.1088/1361-6420/aa9a90.

[4] Elizabeth Newman and Lars Ruthotto. "'hessQuik': Fast Hessian computation of composite functions". In: *Journal of Open Source Software* 7.72 (2022), p. 4171. DOI: 10.21105/joss.04171. URL: https://doi.org/10.21105/joss.04171.

[5] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-