

Physics-Enhanced Neural Networks for Chaotic Hamiltonian Dynamics

Aryan Shrivastava

Roll no. - 2311041

School of Physical Sciences (SPS)

Motivation

Chaotic systems are everywhere.

From molecular dynamics, stormy weather to swirling galaxies etc.

Neural Networks have become popular tools for solving various problems.

We aim to use Neural Networks to predict the dynamics in a Chaotic Hamiltonian System.

Universal Approximation Theorem BUT

Conventional Neural Networks have a "blind spot".

They do not understand that “Clouds are not spheres, mountains are not cones, coastlines are not circles,”

The amount of data and number of neurons it would take would be impractical in the chaotic regime.

Solution

Chaotic systems:





Highly sensitive to initial conditions

We aim to exploit the Hamiltonian structure in the Neural Network.

Need models that **respect physics**, not just the data

We want the Neural Network to learn the "conservation laws"

Physics-enhanced neural networks learn order and chaos

Anshul Choudhary ¹, John F. Lindner ^{1,2,*}, Elliott G. Holliday,¹ Scott T. Miller ¹, Sudeshna Sinha,^{1,3}
and William L. Ditto ¹

¹*Nonlinear Artificial Intelligence Laboratory, Physics Department, North Carolina State University, Raleigh, North Carolina 27607, USA*

²*Physics Department, The College of Wooster, Wooster, Ohio 44691, USA*

³*Indian Institute of Science Education and Research Mohali, Knowledge City, SAS Nagar, Sector 81, Manauli PO 140 306, Punjab, India*



(Received 26 November 2019; revised manuscript received 22 May 2020; accepted 24 May 2020;
published 18 June 2020)

Artificial neural networks are universal function approximators. They can forecast dynamics, but they may need impractically many neurons to do so, especially if the dynamics is chaotic. We use neural networks that incorporate Hamiltonian dynamics to efficiently learn phase space orbits even as nonlinear systems transition from order to chaos. We demonstrate Hamiltonian neural networks on a widely used dynamics benchmark, the Hénon-Heiles potential, and on nonperturbative dynamical billiards. We introspect to elucidate the Hamiltonian neural network forecasting.

DOI: [10.1103/PhysRevE.101.062207](https://doi.org/10.1103/PhysRevE.101.062207)

Project Overview

- Hamiltonian system: **Henon–Heiles**
- Predict the trajectories by 3 methods
 - Artificial Neural Network (ANN)
 - Hamiltonian Neural Network (HNN)
 - Library RK4
- Evaluate their performances by:
 - Phase space Trajectories
 - Energy drift
- Conclusion
- Future Work

Hamiltonian

Hamilton's Equations

$$\dot{q}_i = \frac{\partial \mathcal{H}}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial \mathcal{H}}{\partial q_i}$$

q_i = generalized coordinates, p_i = conjugate momenta, \mathcal{H} = Hamiltonian.

Energy Conservation

$$\frac{d\mathcal{H}}{dt} = 0$$

Total energy is conserved in case of time-symmetry of Hamiltonian.

Hénon-Heiles Hamiltonian

Hamiltonian

$$\mathcal{H}(x, y, p_x, p_y) = \frac{p_x^2 + p_y^2}{2} + \frac{x^2 + y^2}{2} + \left(x^2 y - \frac{y^3}{3} \right)$$

x, y = positions, p_x, p_y = momenta.

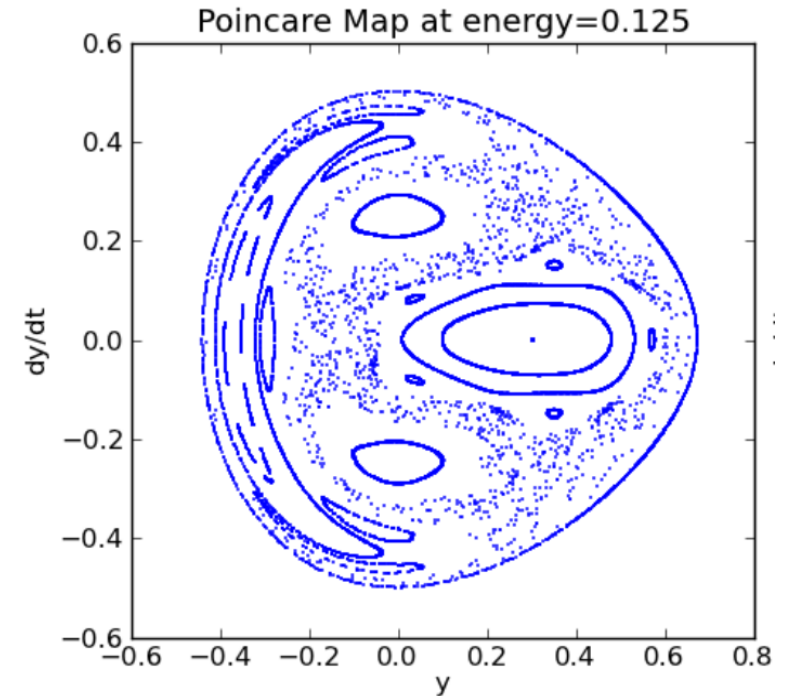
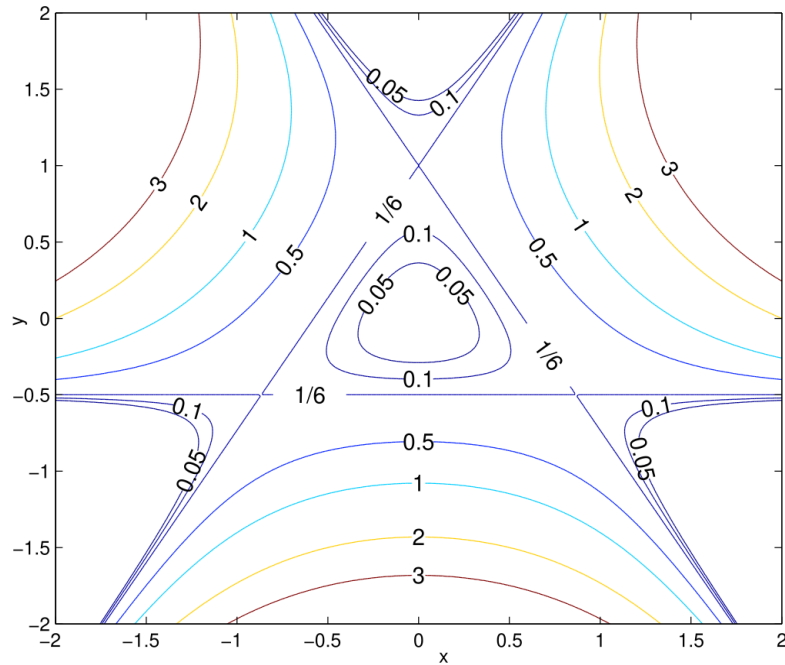
Equations of Motion

$$\begin{aligned}\dot{x} &= p_x, & \dot{y} &= p_y \\ \dot{p}_x &= -x - 2xy \\ \dot{p}_y &= -y - x^2 + y^2\end{aligned}$$

Derived using Hamilton's equations.

Used to model Stellar dynamics and Molecular dynamics

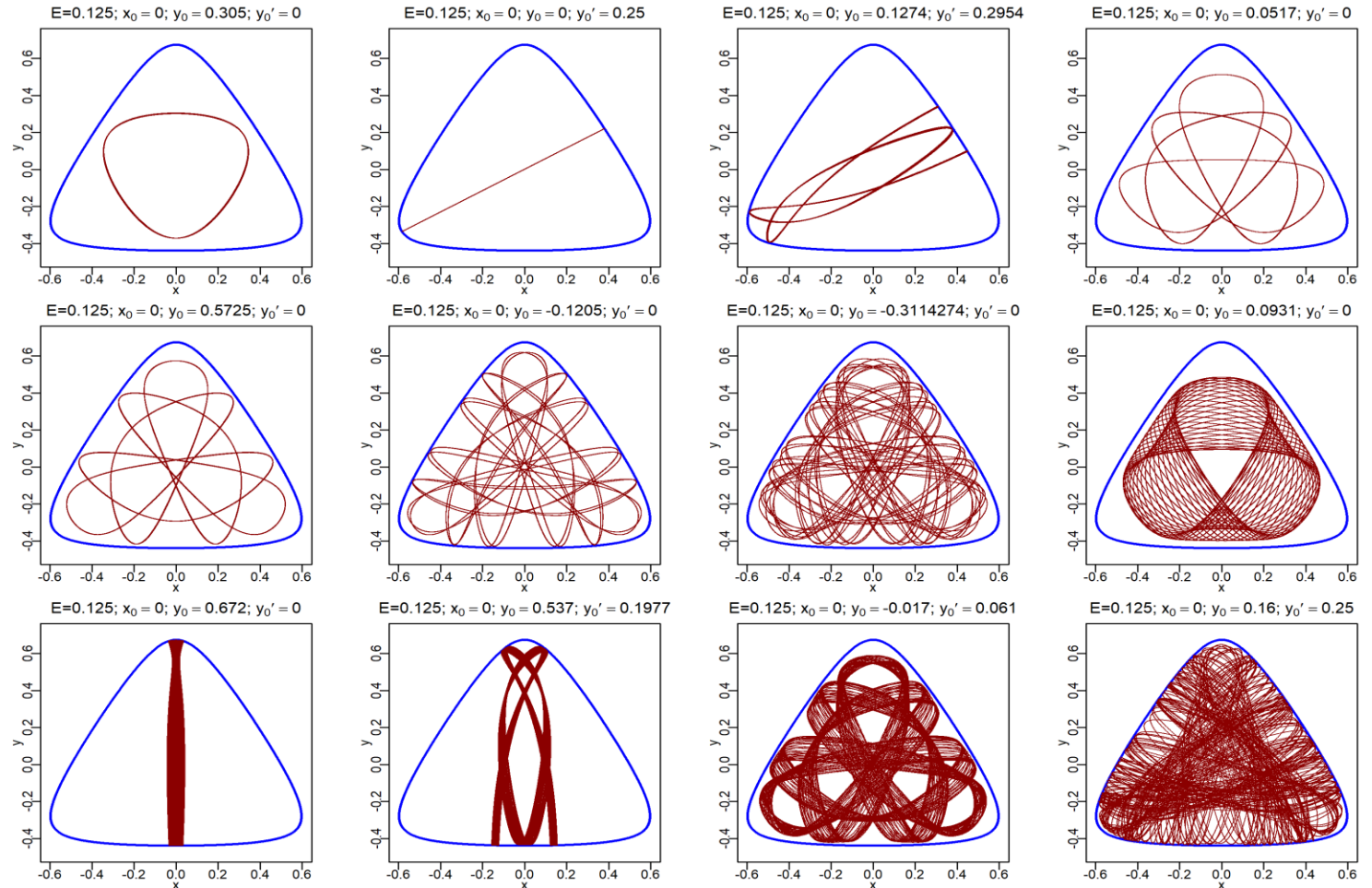
Hénon-Heiles Hamiltonian



Bounded motion is possible in a triangular region of the $\{x, y\}$ plane for energies $0 < E < 1/6$.

Hénon-Heiles Hamiltonian

Chaos and Order coexist for same energies for different initial conditions



- Adapted from a blog post

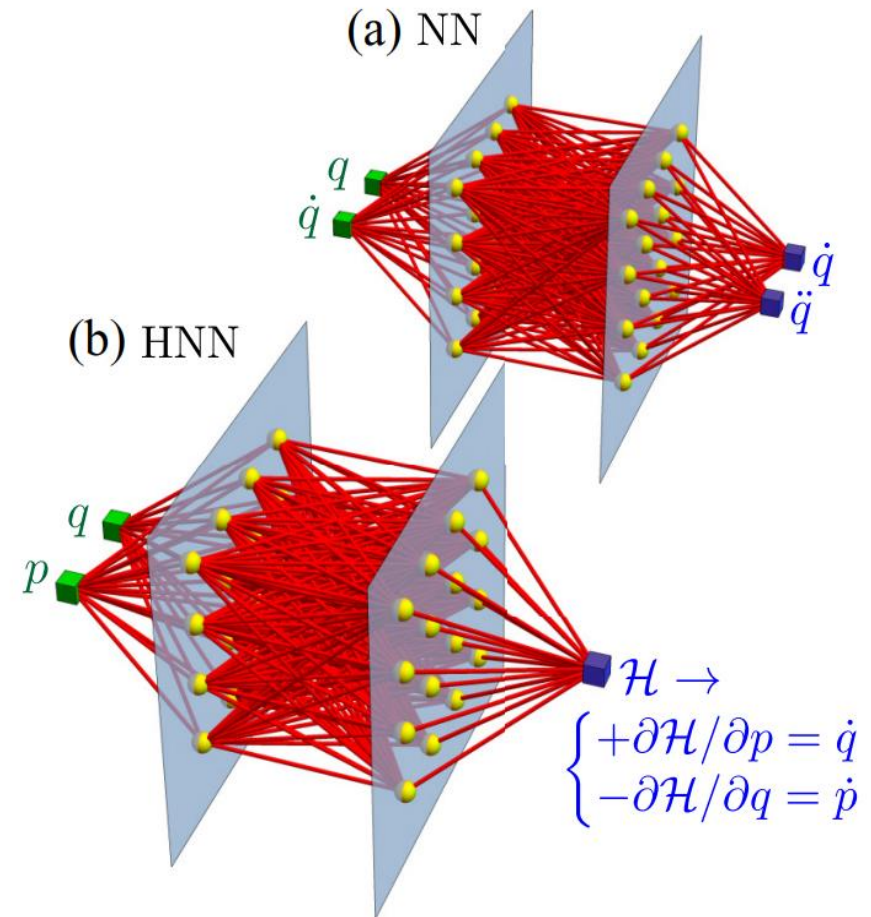
Dataset & Model Architecture

Integrate the Henon-Heiles using
`scipy.integrate.solve_ivp` (Uses RK45)

For energies: $0 < E < 1/6$ (bounded trajectories)

We use conventional feed-forward neural networks

Model	ANN	HNN
Input	$\{q, \dot{q}\}$	$\{q, p\}$
Output	$\{\dot{q}, \ddot{q}\}$	H (scalar Hamiltonian)



- Adapted from the reference paper

Dataset & Model Architecture

Orbit time in the dataset used was 100. (due to large training time)

While making predictions, we made predictions till orbit time = 1800.

TABLE I. Python neural network parameters. We explored all values but generated our final results with the **bold** ones.

Description	Values
Number of layers	2, 4, 6, 8
Neurons per layer	100, 200 , 400
Optimizer	Adam , SGD
Training loss threshold	10^{-4} , 10^{-5}
Batch size	128, 256 , 512
Epochs	10, 50 , 100
Activations	Tanh , ReLU
Orbit time	1000, 5000
Energy samples	20
Orbits per energy	20 , 50
Integration scheme	RK4, RK45 , Symplectic
Data sampling rate	0.01, 0.1 , 1, 10

- Adapted from the reference paper

Dataset & Model Architecture

Model: "ANN"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 4)	0
dense (Dense)	(None, 200)	1,000
dense_1 (Dense)	(None, 200)	40,200
dense_2 (Dense)	(None, 4)	804

Total params: 42,004 (164.08 KB)

Trainable params: 42,004 (164.08 KB)

Non-trainable params: 0 (0.00 B)

Artificial Neural Network (ANN) Loss

$$L_{\text{ANN}} = \|\dot{q}^t - \dot{q}\|^2 + \|\dot{p}^t - \dot{p}\|^2$$

(\dot{q}^t, \dot{p}^t) = target derivatives; (\dot{q}, \dot{p}) = NN predictions.

Model: "HNN"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 4)	0
dense_3 (Dense)	(None, 200)	1,000
dense_4 (Dense)	(None, 200)	40,200
dense_5 (Dense)	(None, 1)	201

Total params: 41,401 (161.72 KB)

Trainable params: 41,401 (161.72 KB)

Non-trainable params: 0 (0.00 B)

Hamiltonian Neural Network (HNN) Loss

$$L_{\text{HNN}} = \left\| \dot{q}^t - \frac{\partial \mathcal{H}}{\partial p} \right\|^2 + \left\| \dot{p}^t + \frac{\partial \mathcal{H}}{\partial q} \right\|^2$$

HNN outputs scalar \mathcal{H} ; its gradients are used to find \dot{q} and \dot{p}

Dataset & Model Architecture

Euler update for ANN:

$$(q, p)_{t+\Delta t} = (q, p)_t + (\dot{q}, \dot{p}) \Delta t$$

Tried using RK4, and other methods,
but as suggested in the paper, Euler is
the most practical.

(This is a very slow for other methods)

Using Library RK4 for HNN update
(Reference paper to used RK45)
(not slow)

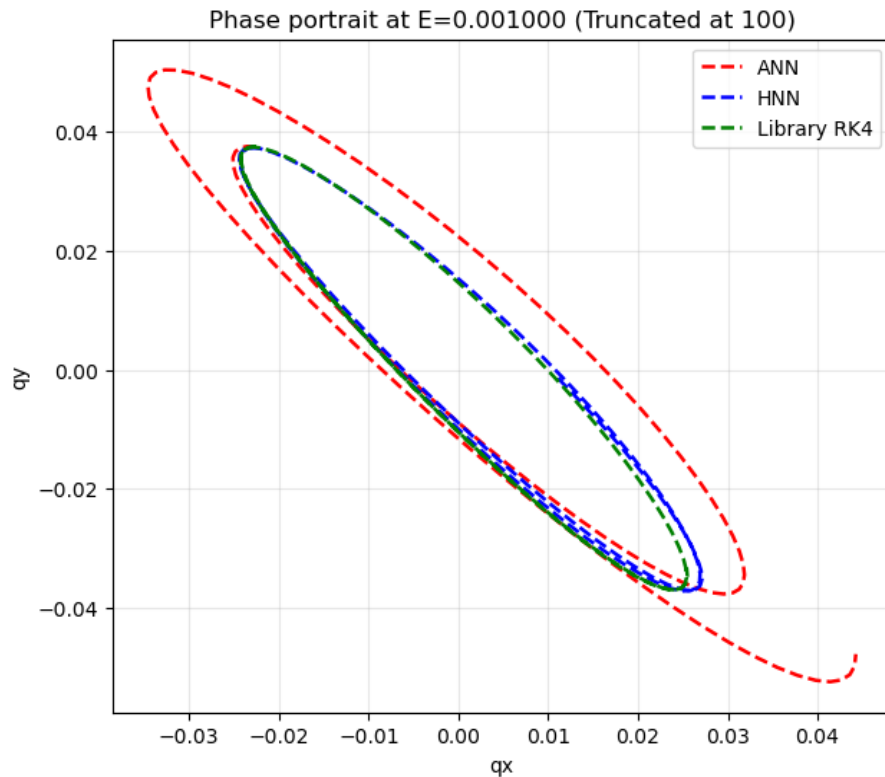
TABLE I. Python neural network parameters. We explored all values but generated our final results with the **bold** ones.

Description	Values
Number of layers	2 , 4, 6, 8
Neurons per layer	100, 200 , 400
Optimizer	Adam , SGD
Training loss threshold	10^{-4} , 10^{-5}
Batch size	128, 256 , 512
Epochs	10, 50 , 100
Activations	Tanh , ReLU
Orbit time	1000, 5000
Energy samples	20
Orbits per energy	20 , 50
Integration scheme	RK4, RK45 , Symplectic
Data sampling rate	0.01, 0.1 , 1, 10

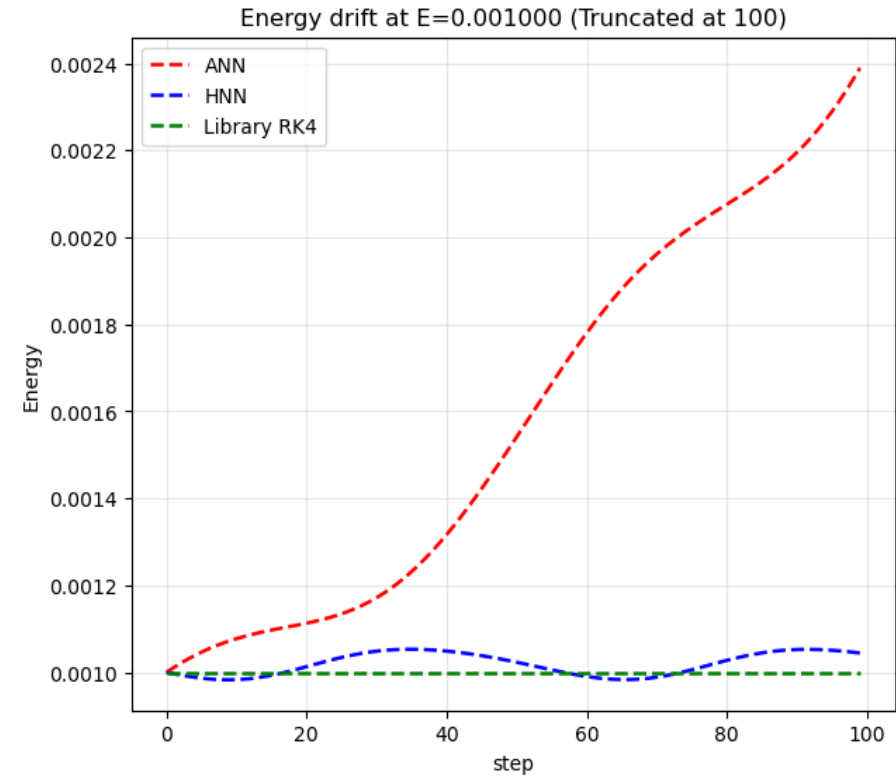
- Adapted from the reference paper

Trajectories and Energy drift

ANN Trajectories diverge after around 100 steps

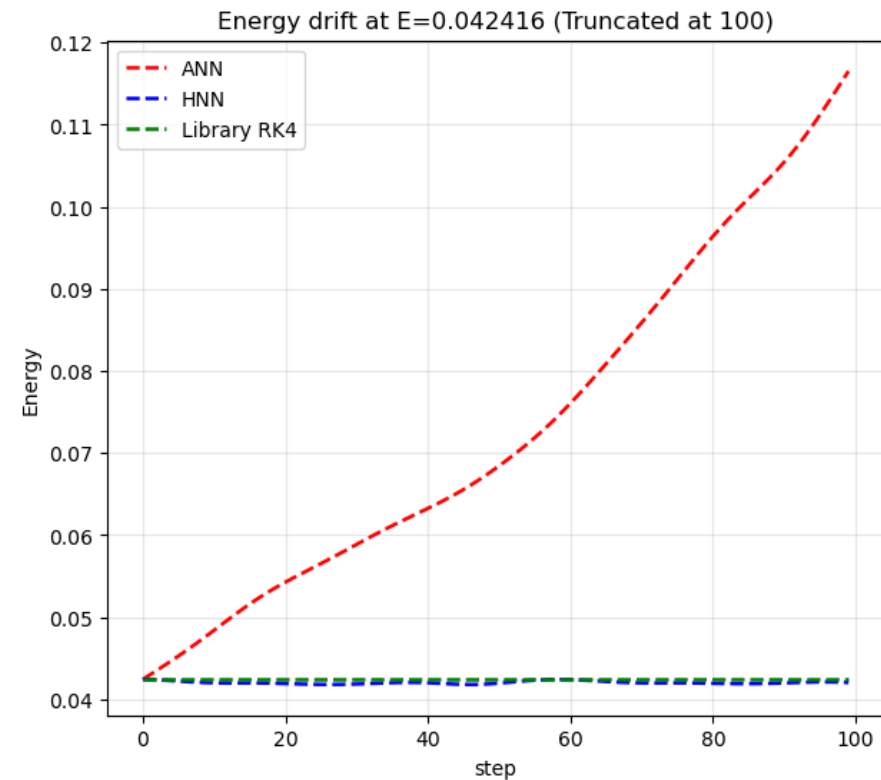
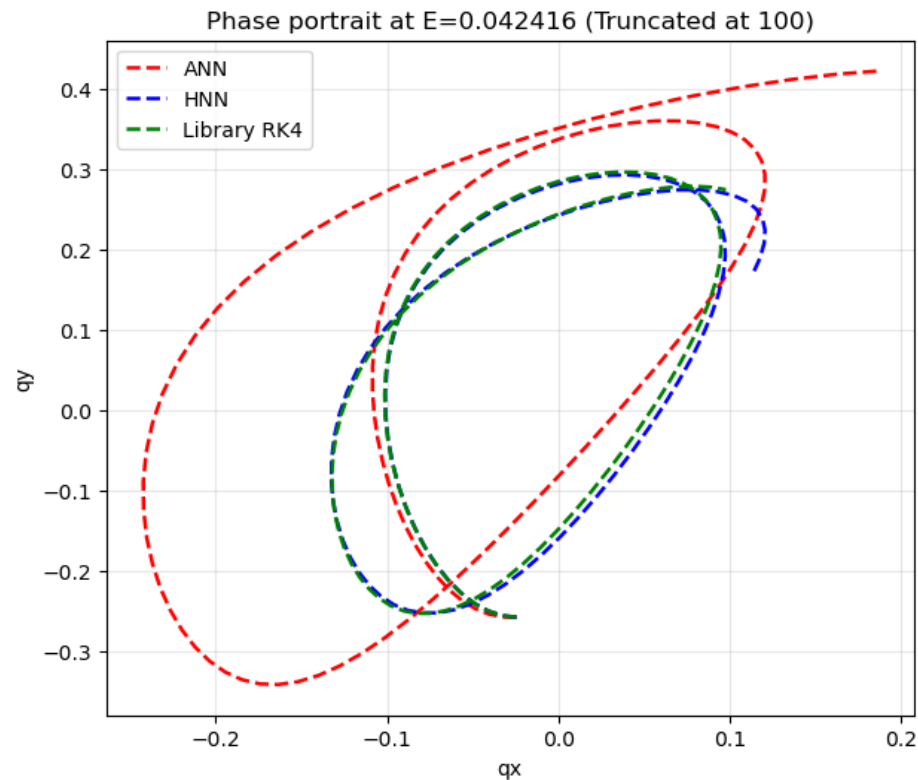


ANN doesn't conserve energy



Trajectories and Energy drift

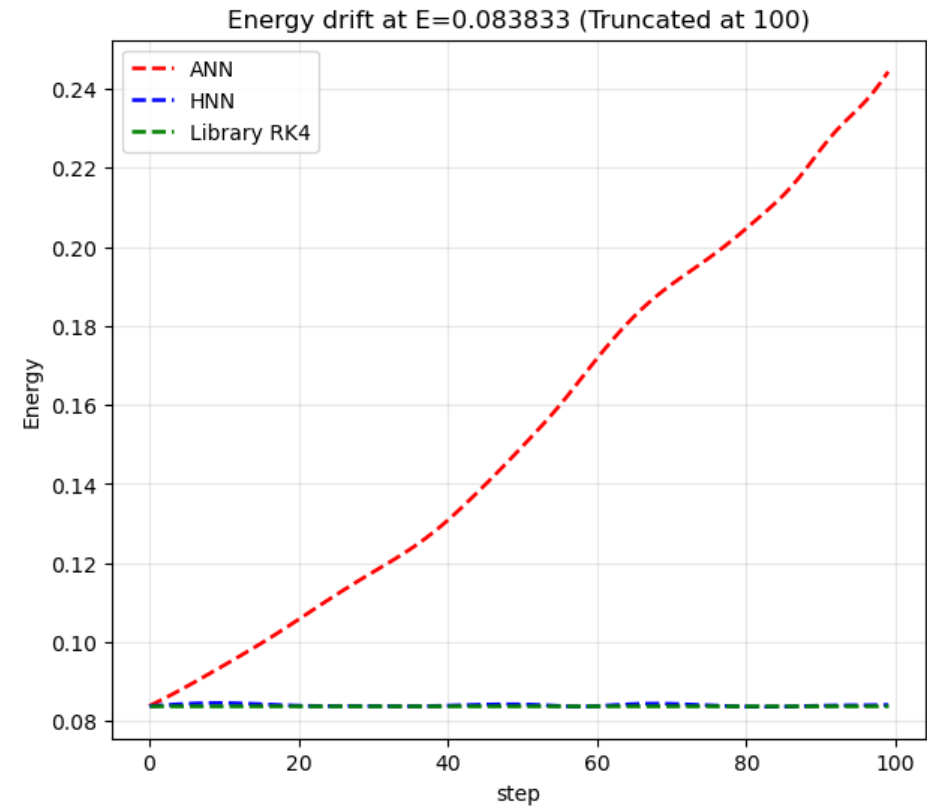
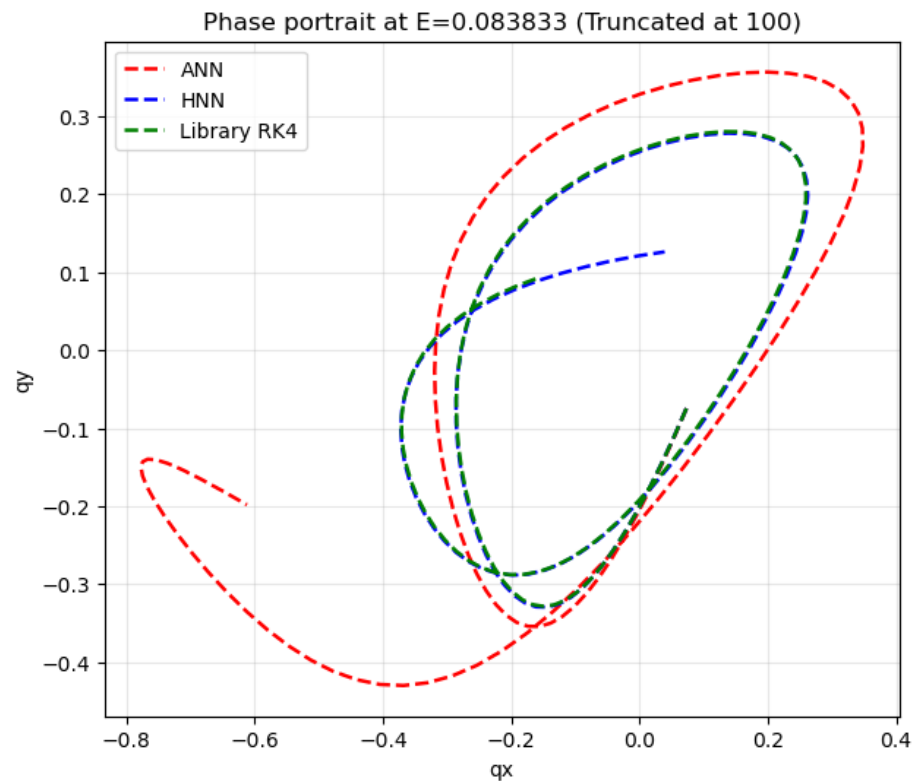
Tried for random initial conditions at different energies from 0 to $1/6$.



Trajectories and Energy drift

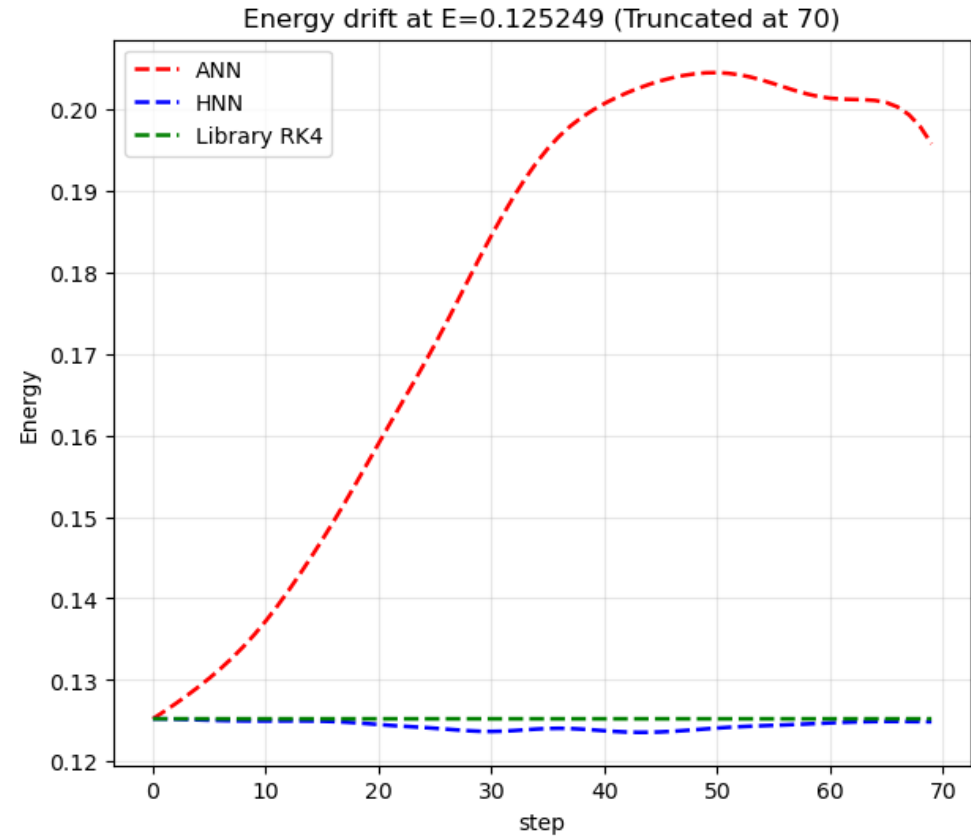
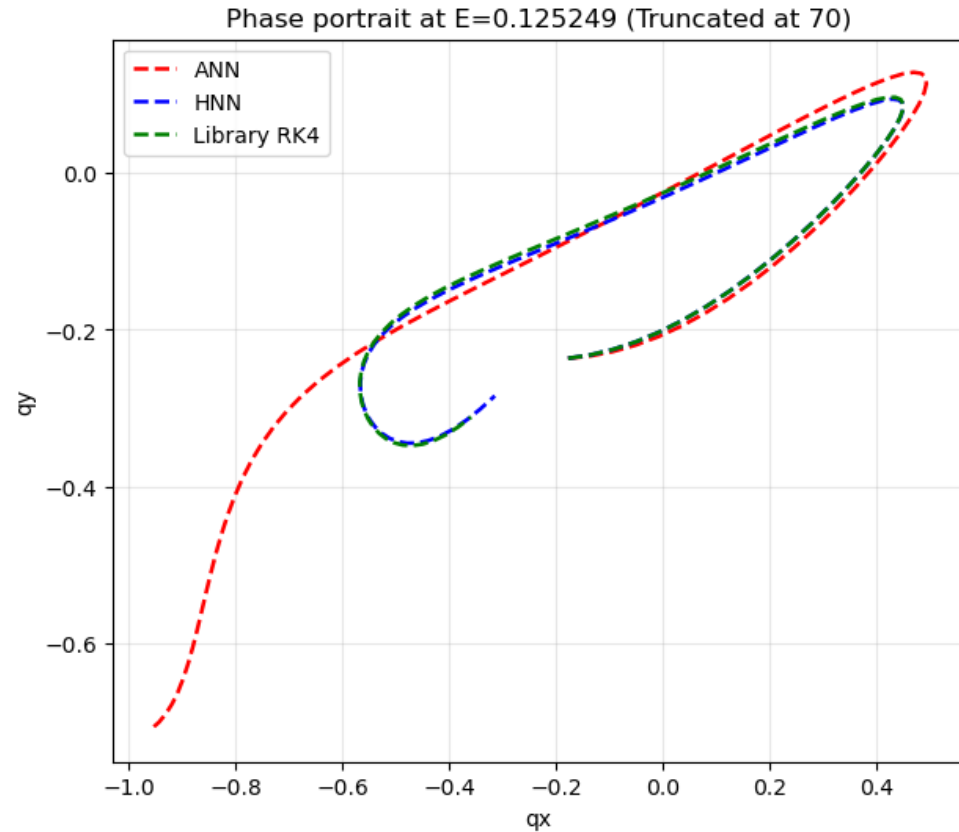
Got the same results for all energies.

ANN -> diverges, HNN -> bounded (close to Library RK4)



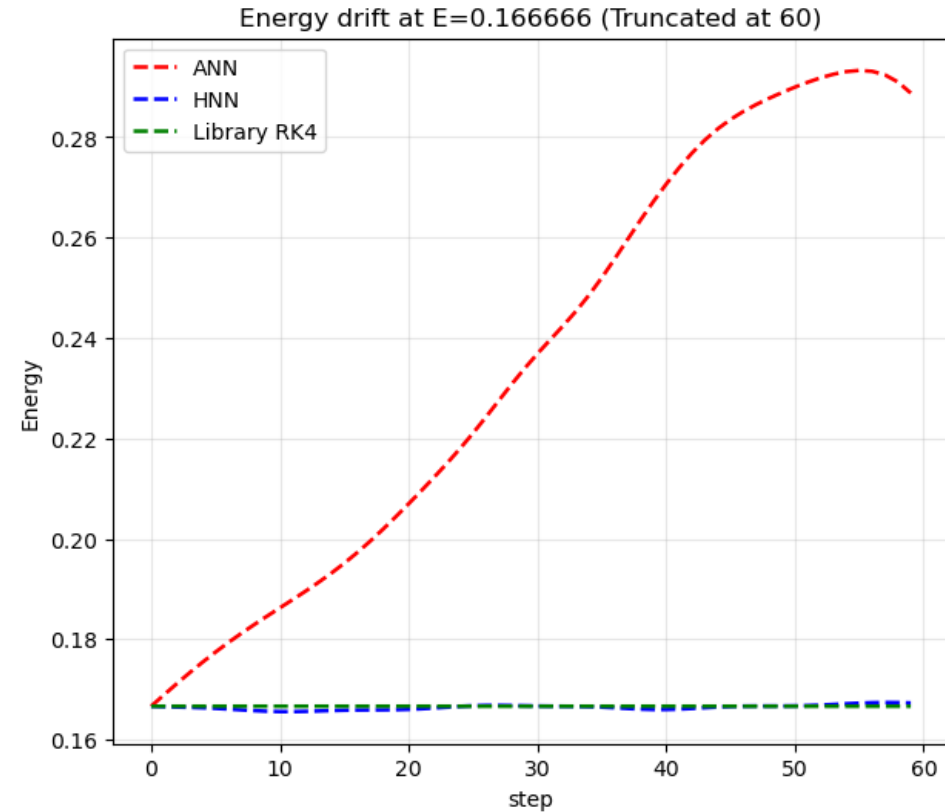
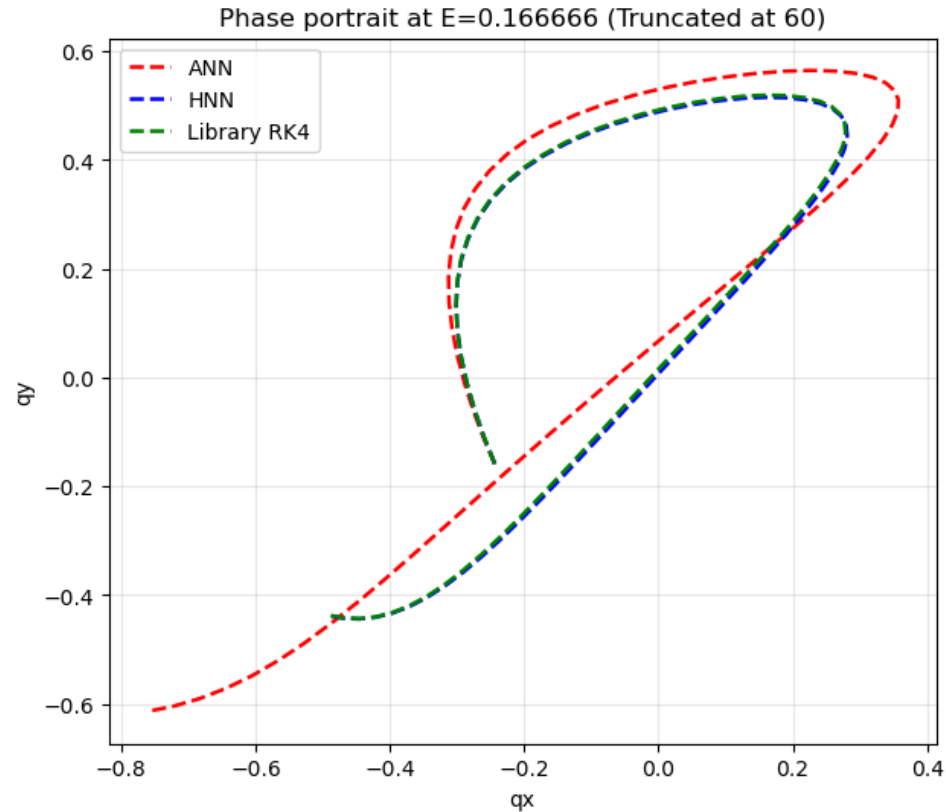
Trajectories and Energy drift

At higher energies, ANN diverges faster. (70 time steps)



Trajectories and Energy drift

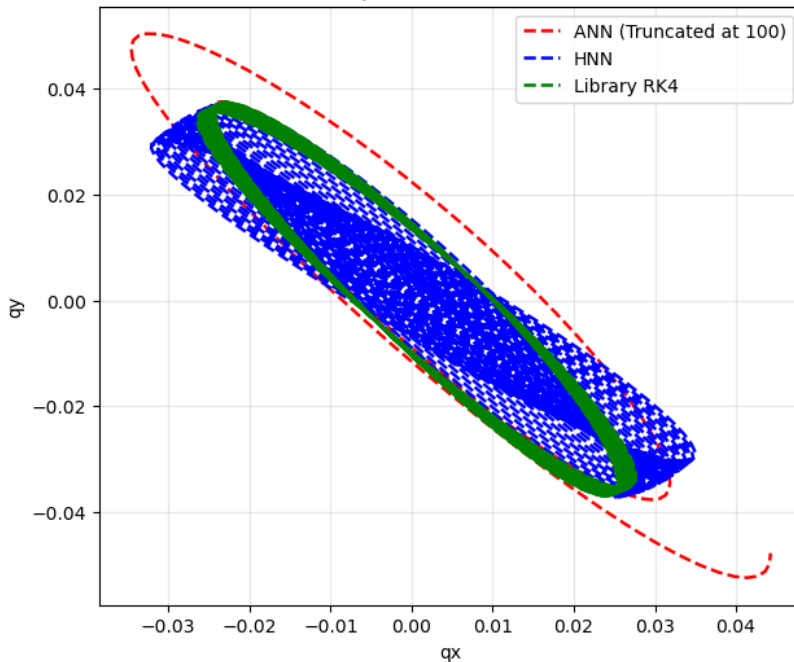
At highest energy for bounded orbits. $E = 1/6$, ANN diverges only after 60 time steps



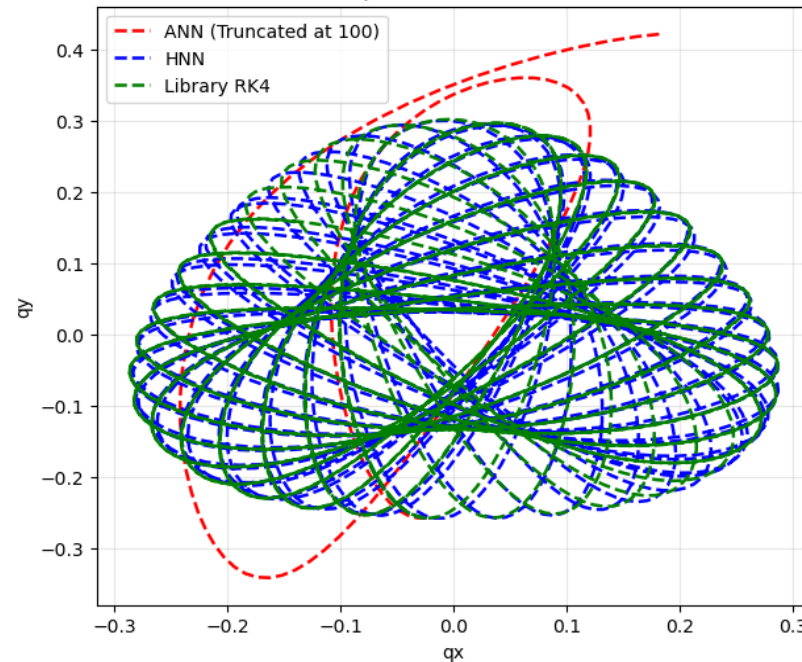
Trajectories and Energy drift

We plot the full trajectories of HNN and Library RK4, to show that they do not diverge after long time steps too. Orbit time = 1800 (reduced compared to the paper's 5000 orbit time to reduce computational time).

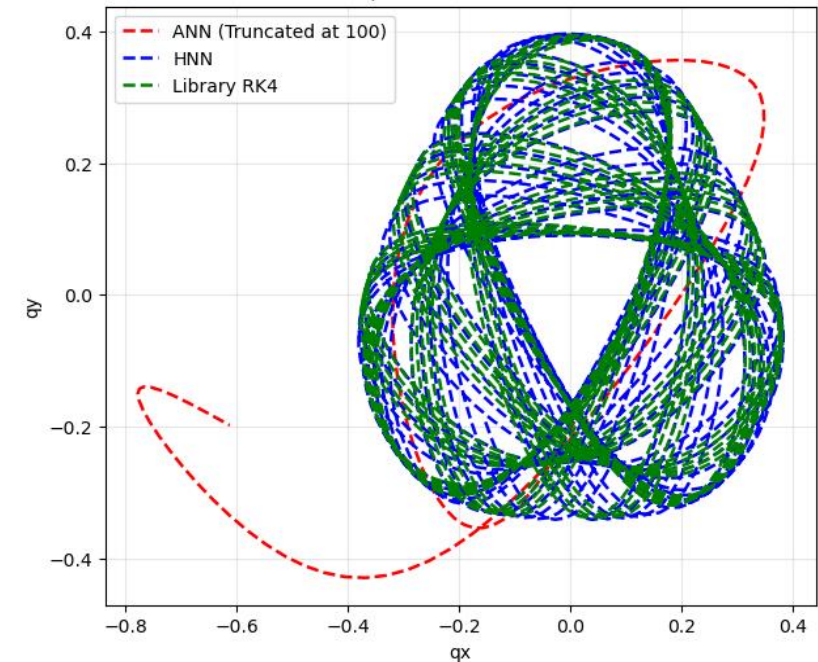
Phase portrait at $E=0.001000$



Phase portrait at $E=0.042416$

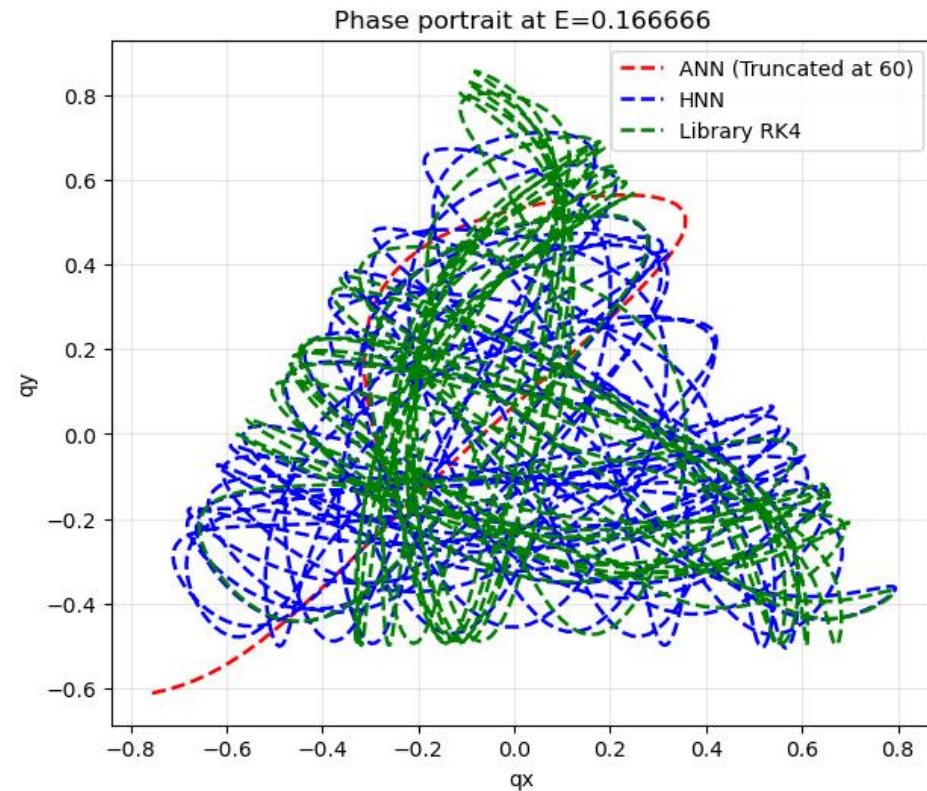
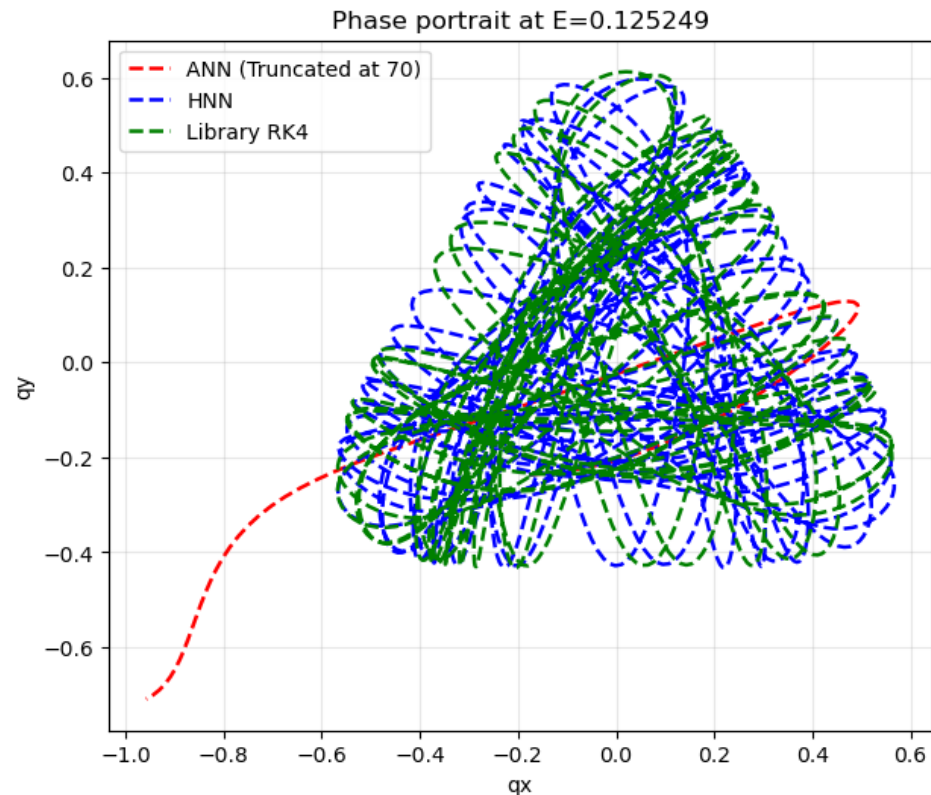


Phase portrait at $E=0.083833$



Trajectories and Energy drift

Even at higher energies, HNN doesn't diverge. It shows that the HNN has learned the dynamics of the system.

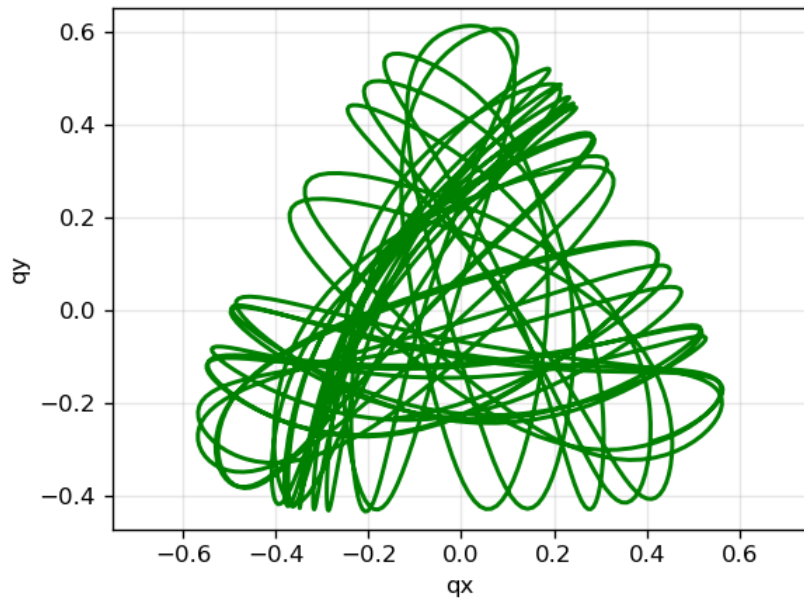


Side by side comparison

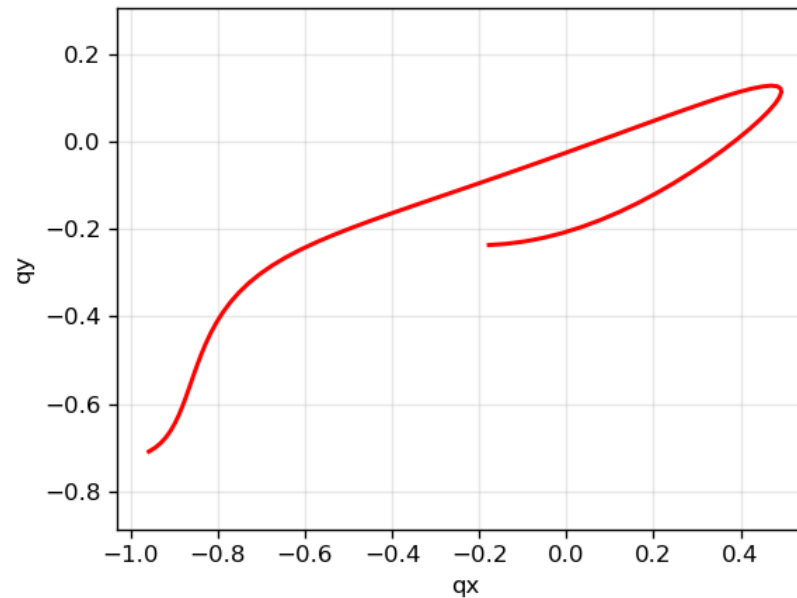
For Energy $E = 0.125249$

ANN diverged after 70 steps

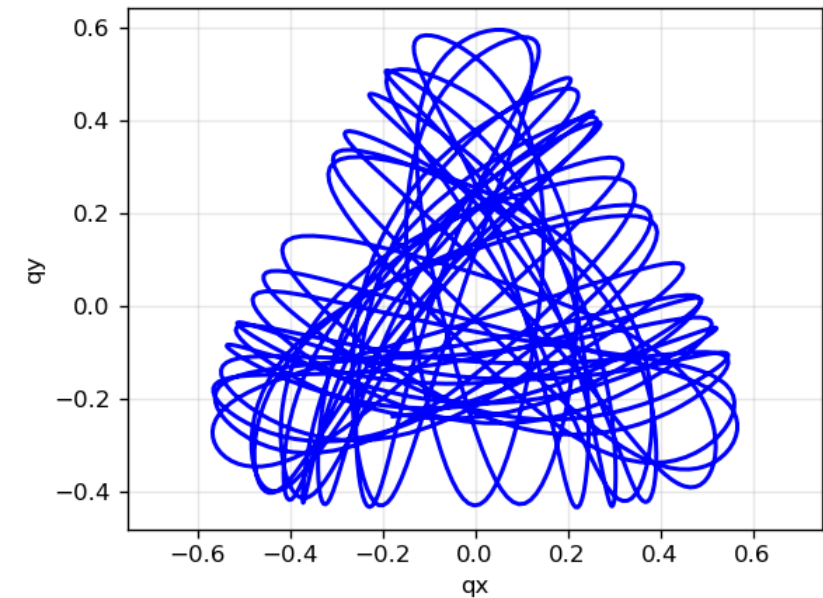
Library RK4 Trajectory



ANN Trajectory (0:70)



HNN Trajectory

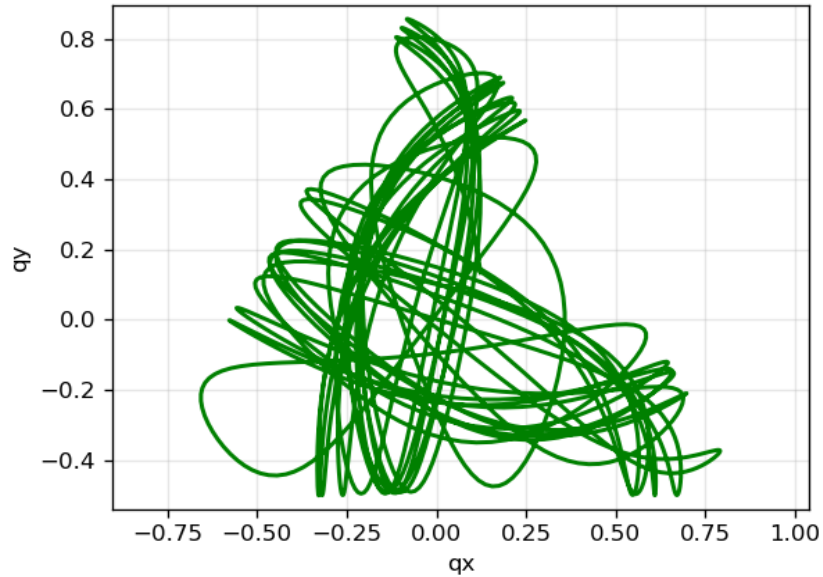


Side by side comparison

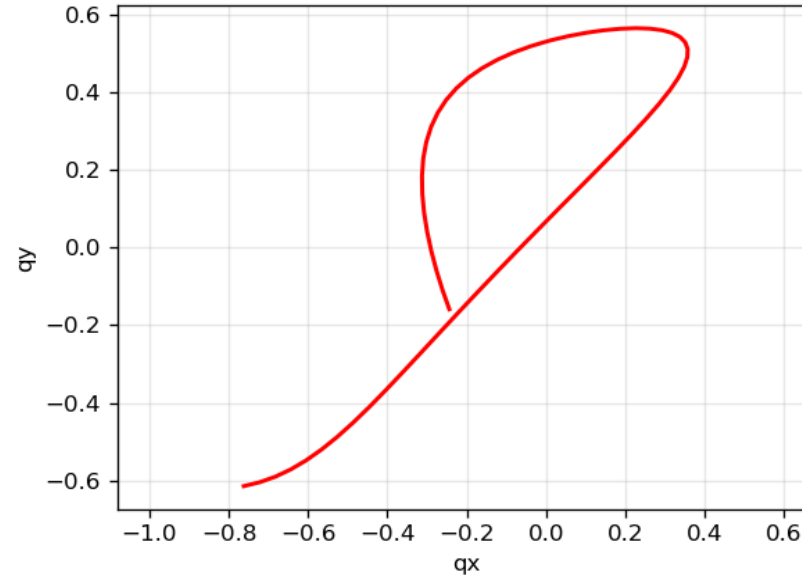
For Energy $E = 0.166666$ (max energy for bounded trajectories)

ANN diverged after 60 steps

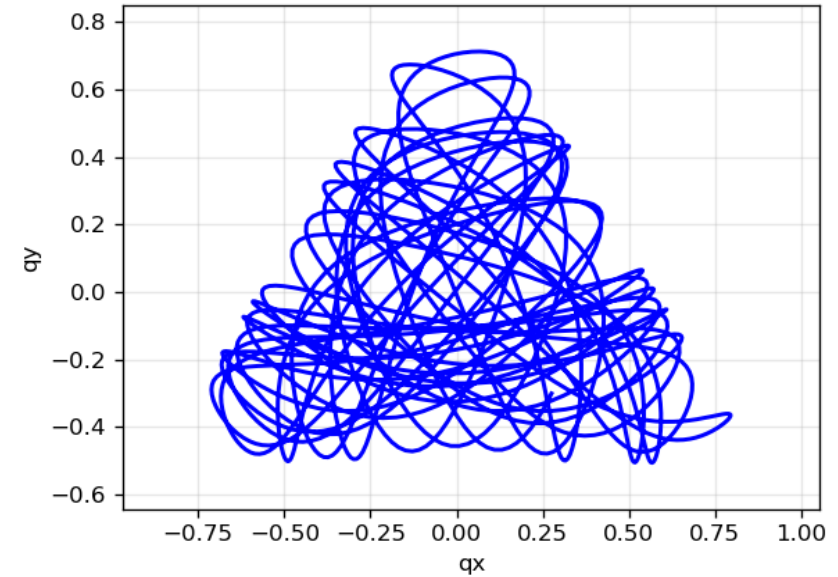
Library RK4 Trajectory



ANN Trajectory (0:60)



HNN Trajectory



Conclusion

ANNs learned the trajectories for given initial conditions.

HNNs learned the vector field (since it was trained for orbit time = 100, and still it gave reasonable trajectory for orbit time = 1800).

HNNs overcome the "**chaos-blindness**"

By using energy-conserving flows, coming from the Hamiltonian, **without invoking any details of its form**, HNNs outperform the traditional ANNs.

That means, we can use this idea for other Hamiltonian systems

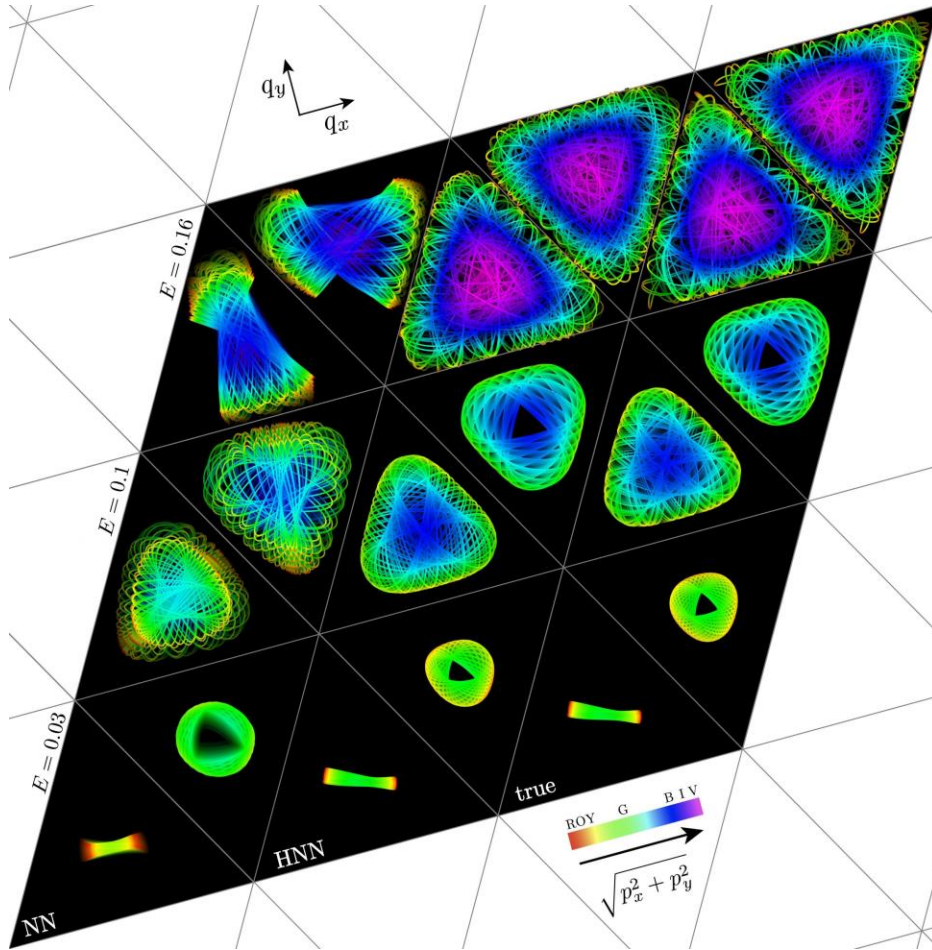
Future Work

Use more quantitative measures to evaluate model performances like Lyapunov exponent.

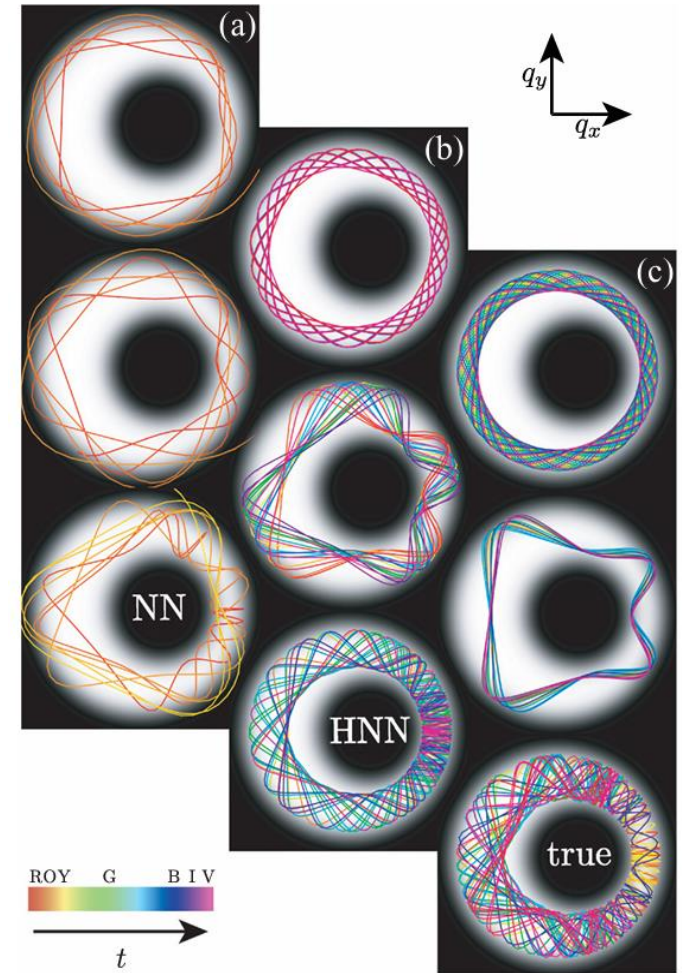
Apply the idea to Hamiltonian systems like double pendulum.

Lagrangian Neural Networks for non-conservative systems (conserve phase-space volume, but not energy).

Reference paper's results



Billiards Table
With soft walls



Thank You

Questions?