

Structure-Preserving Neural Networks for Hamiltonian Systems

GACM Colloquium on Computational Mechanics 2022

Philipp Horn and Barry Koren (Eindhoven University of Technology)
Veronica Saz Ulibarrena and Simon Portegies Zwart (Leiden Observatory)

What is a Hamiltonian System?

The ODE describing a Hamiltonian system can be written as:

$$\begin{aligned} \begin{pmatrix} \dot{p} \\ \dot{q} \end{pmatrix} &= J r H(p; q); \quad \text{with } J = \begin{pmatrix} 0 & I_d \\ I_d & 0 \end{pmatrix} \\ &= \begin{pmatrix} r_q H(p; q) \\ r_p H(p; q) \end{pmatrix}; \quad p, q \in \mathbb{R}^d; \end{aligned} \quad (1)$$

In some Hamiltonian systems the Hamiltonian $H: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is separable, meaning:

$$H(p; q) = T(p) + U(q); \quad (2)$$

Then, $T: \mathbb{R}^d \rightarrow \mathbb{R}$ is referred to as the kinetic energy and $U: \mathbb{R}^d \rightarrow \mathbb{R}$ as the potential energy.

Two key properties of Hamiltonian systems

1. Total energy is conserved along trajectories:

$$H(p_0; q_0) = H(p(t); q(t)); \quad t \in \mathbb{R}_+ \quad (3)$$

where $(p(t); q(t))^T$ is the solution of ODE (1) with initial condition $(p_0; q_0)^T$

2. Symplecticity of the flow map $\Phi_h : x(t) \mapsto x(t+h)$ meaning:

$$\left(\frac{\partial \Phi_h}{\partial x} \right)^T J \frac{\partial \Phi_h}{\partial x} = J; \quad x = \begin{pmatrix} p \\ q \end{pmatrix} \in \mathbb{R}^{2d}; \quad (4)$$

How to solve a Hamiltonian system

Classically one knows the Hamiltonian H of the system and using (symplectic) numerical integrators one can solve Hamilton's equations. Alternatively, one can use a neural network trained on data

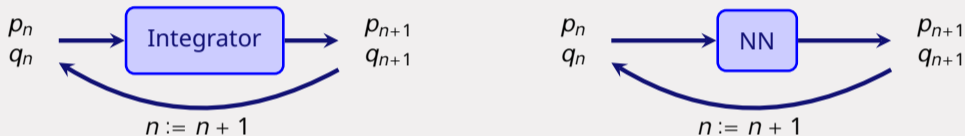


Figure: Iterative scheme to solve an ODE using a numerical integrator and a neural network.

Semi-implicit (symplectic) Euler method

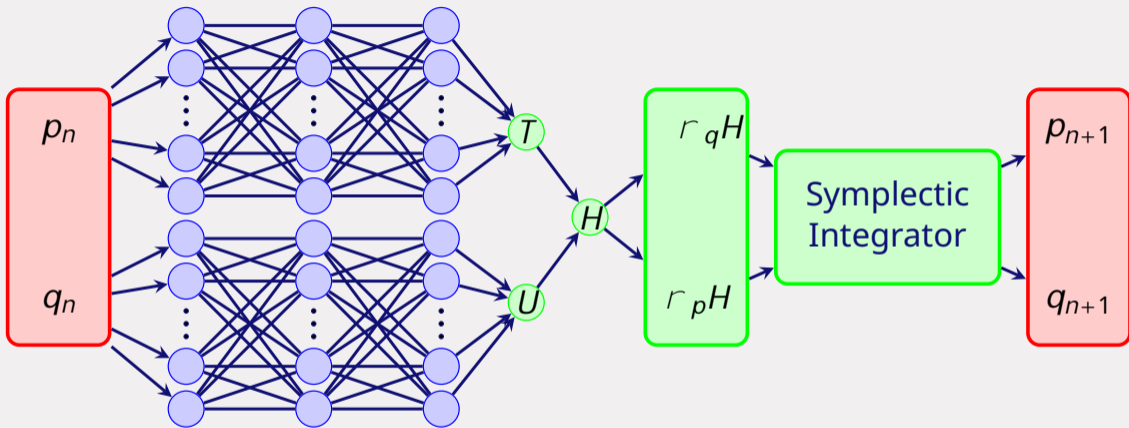
The symplectic Euler method for Hamiltonian systems given p_n and q_n is:

$$\begin{aligned} p_{n+1} &= p_n - h r_q H(p_{n+1}; q_n) \\ q_{n+1} &= q_n + h r_p H(p_{n+1}; q_n) \end{aligned}$$

For separable Hamiltonian systems it becomes an explicit method:

$$\begin{aligned} x_n := \begin{pmatrix} p_n \\ q_n \end{pmatrix} & \begin{matrix} \text{!} \\ \end{matrix} \begin{pmatrix} p_n - r U(q_n)h \\ q_n \end{pmatrix} & =: \begin{pmatrix} p_{n+1} \\ q_n \end{pmatrix} \text{!} \\ & \begin{matrix} \text{!} \\ \end{matrix} \begin{pmatrix} p_{n+1} \\ q_n + r T(p_{n+1})h \end{pmatrix} & =: \begin{pmatrix} p_{n+1} \\ q_{n+1} \end{pmatrix} =: x_{n+1} \end{aligned} \quad (5)$$

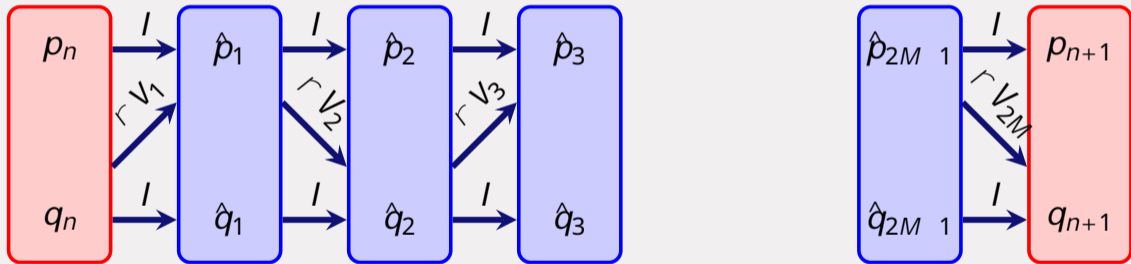
Hamiltonian neural network by Chen *et al.* [1]



[1] Z. Chen *et al.*, Symplectic recurrent neural networks, *8th International Conference on Learning Representations, ICLR, 2020.*

A standard SympNet by Jin *et al.* [2]

- Each layer: $\mathbb{R}^{2d} \rightarrow \mathbb{R}^{2d}$ and is a symplectic map
- A concatenation of symplectic maps is again symplectic) A SympNet is a symplectic map.



[2] P. Jin *et al.*, SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems, *Neural Networks*, Vol. **132**, pp. 166-179, 2020.

SympNets - Gradient layers

A gradient layer has the trainable parameters: $K \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$ and $a \in \mathbb{R}^n$. Also, an activation function has to be chosen beforehand. Usually the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ is used. With this (and a slight abuse of matrix vector multiplication notation) the upper and lower gradient layers are defined as:

$$G_{up} \begin{pmatrix} p \\ q \end{pmatrix} := \begin{pmatrix} I & \hat{\Lambda}_{K,a,b} \\ 0 & I \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p + K^T \text{diag}(a) (Kq + b) \\ q \end{pmatrix} \quad (6)$$

$$G_{low} \begin{pmatrix} p \\ q \end{pmatrix} := \begin{pmatrix} I & 0 \\ \hat{\Lambda}_{K,a,b} & I \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p \\ q + K^T \text{diag}(a) (Kp + b) \end{pmatrix} \quad (7)$$

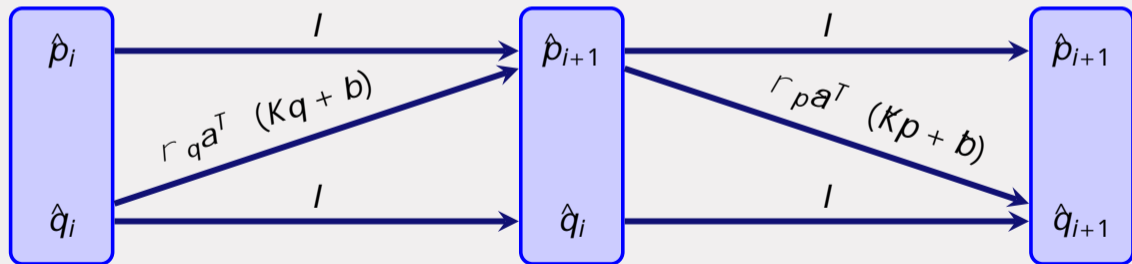
The name of this layer comes from the fact that $\hat{\Lambda}_{K,a,b}$ can approximate any gradient of a function $V: \mathbb{R}^d \rightarrow \mathbb{R}$ (so any $r \in \mathbb{R}^d$).

Two symplectic gradient layers in detail

Focusing on two alternating gradient layers and using

$$r_q a^T (Kq + b) = K^T \text{diag}(a) (Kq + b) \quad (8)$$

$$r_p a^T (Kp + b) = K^T \text{diag}(a) (Kp + b) \quad (9)$$



Two gradient layers learn a Hamiltonian

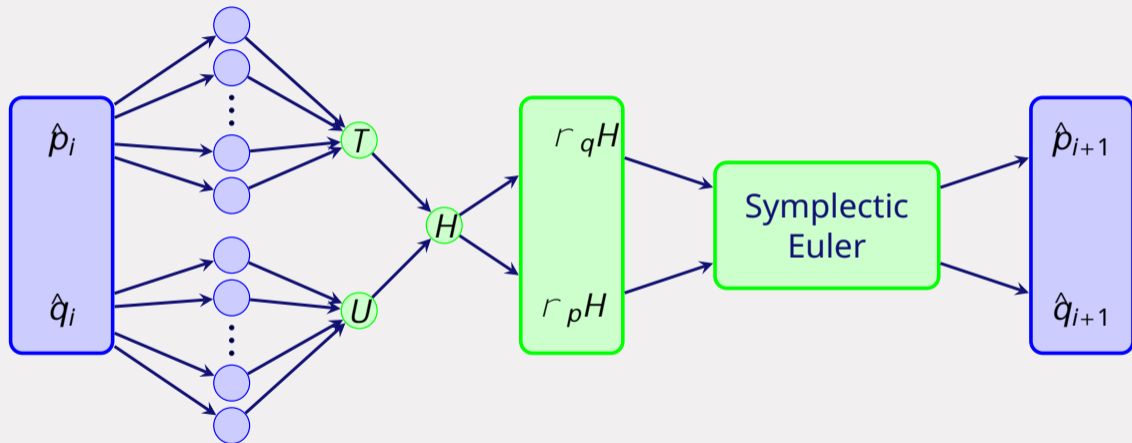
Two layers, starting with an upper and followed by a lower gradient layer can be written as:

$$\begin{array}{l} p_i \\ q_i \end{array} \begin{array}{l} | \\ | \end{array} \begin{array}{l} p_i + r_q a^T (Kq_i + b) \\ q_i \end{array} =: \begin{array}{l} p_{i+1} \\ q_i \end{array} \begin{array}{l} | \\ | \end{array} \\ \begin{array}{l} | \\ | \end{array} \begin{array}{l} p_{i+1} \\ q_i + r_p a^T (Kp_{i+1} + b) \end{array} =: \begin{array}{l} p_{i+1} \\ q_{i+1} \end{array} \quad (10)$$

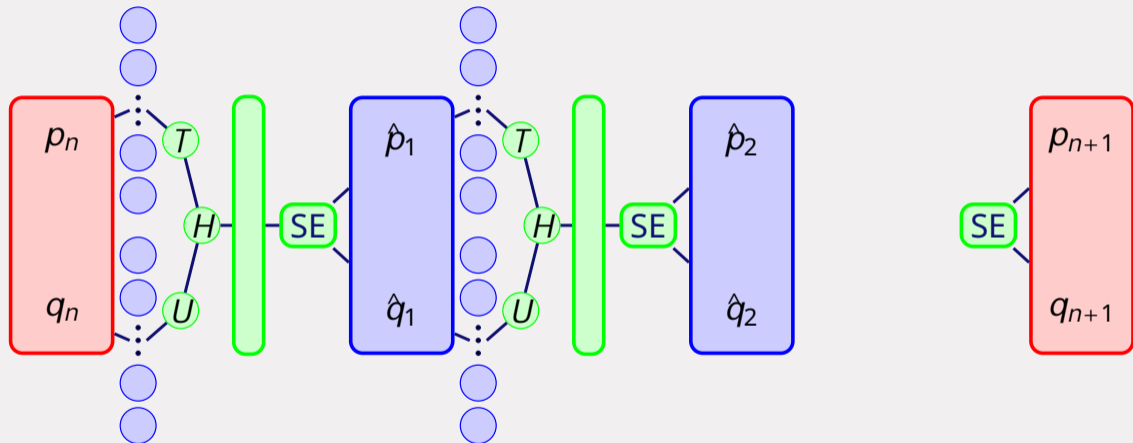
These are the updates of the symplectic Euler method for the Hamiltonian

$$H(p; q) = \frac{1}{h} \underbrace{a^T (Kp + b)}_{T(p)} \quad \frac{1}{h} \underbrace{a^T (Kq + b)}_{U(q)} \quad (11)$$

Two gradient layers as an HNN

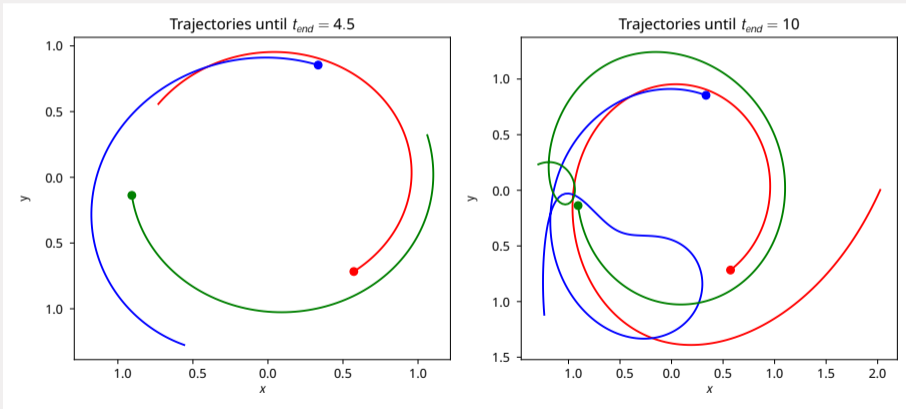


SympNet as multiple HNNs in series



Data for numerical experiments

- Three bodies with equal masses
- Disturbed circular orbit
- In the x - y -plane
- 5000 trajectories until $t_{\text{end}} = 10$



SympNets vs physics-unaware neural networks

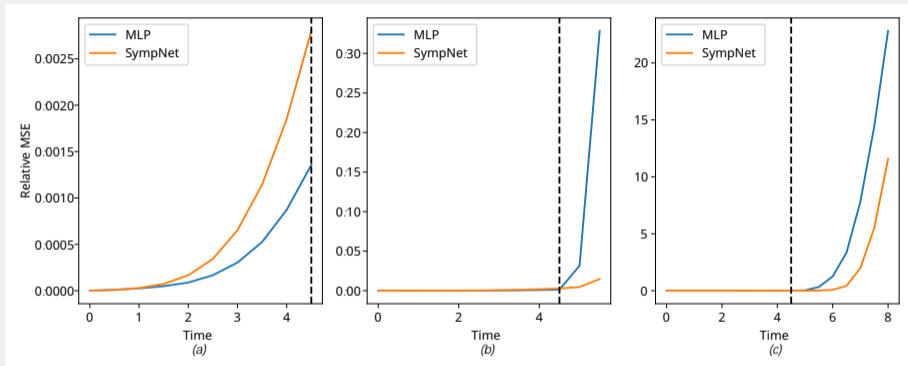
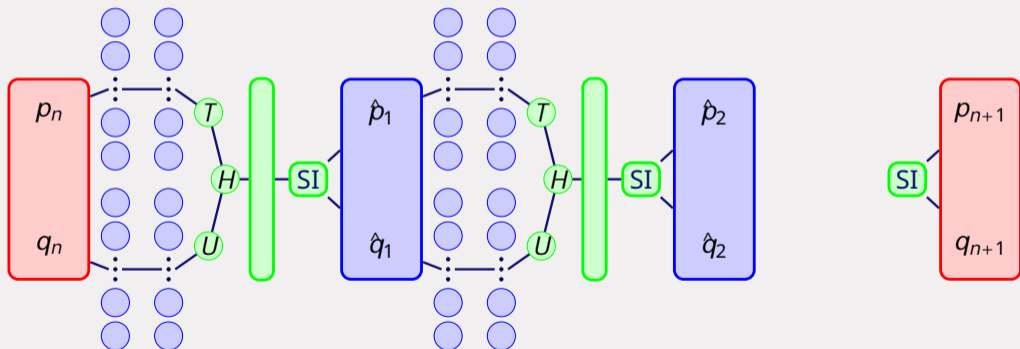


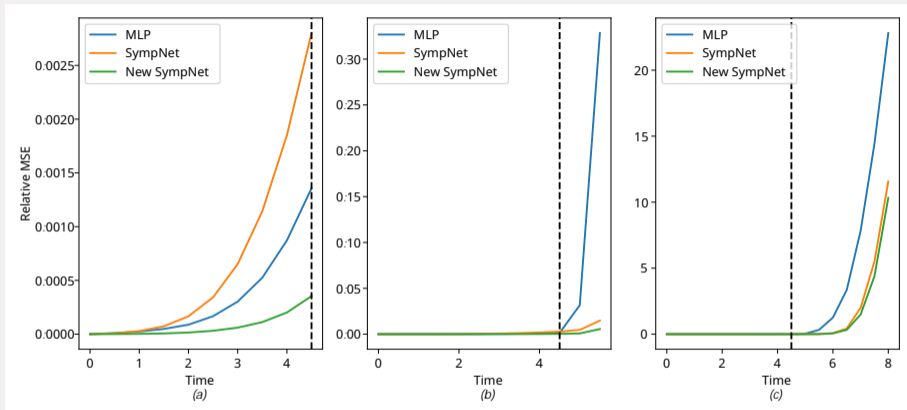
Figure: Relative mean square error (MSE) for a physics-unaware multilayer perceptron (MLP) and a SympNets trained on data until $t_{\text{end}} = 4.5$ (vertical line).

Improved SympNets

The new point of view on the topology of SympNets allows us to introduce a generalized type of symplectic neural networks. SympNets and HNNs are only two special cases of these NNs.



Improved SympNets



Conclusion

- We have shown a new connection between SympNets and HNNs
- Physics-unaware MLPs are more accurate inside the training data than SympNets
- SympNets generalize better than physics-unaware MLPs

Outlook

- A more generalized version of SympNets can be constructed
- Improved SympNets can outperform physics-unaware MLPs inside the training data and SympNets outside the data