The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

STRUCTURE-PRESERVING NEURAL NETWORKS FOR THE N-BODY PROBLEM ECCOMAS CONGRESS 2022

PHILIPP HORN[1], VERONICA SAZ ULIBARRENA[2], BARRY KOREN[1] AND SIMON PORTEGIES ZWART[2]

[1] Centre for Analysis, Scientific Computing and Applications (CASA)
Eindhoven University of Technology
PO Box 513, 5600 MB Eindhoven, Netherlands
e-mail: p.horn@tue.nl, b.koren@tue.nl

[2] Leiden Observatory
Leiden University
PO Box 9513, 2300 RA Leiden, Netherlands
e-mail: ulibarrena@strw.leidenuniv.nl, spz@strw.leidenuniv.nl

Key words: Neural Networks, Structure-Preserving Computing, Symplectic Algorithms, Astrophysics

Abstract. In order to understand when it is useful to build physics constraints into neural networks, we investigate different neural network topologies to solve the N-body problem. Solving the chaotic N-body problem with high accuracy is a challenging task, requiring special numerical integrators that are able to approximate the trajectories with extreme precision. In [1] it is shown that a neural network can be a viable alternative, offering solutions many orders of magnitude faster. Specialized neural network topologies for applications in scientific computing are still rare compared to specialized neural networks for more classical machine learning applications. However, the number of specialized neural networks for Hamiltonian systems has been growing significantly during the last years [3, 5]. We analyze the performance of SympNets introduced in [5], preserving the symplectic structure of the phase space flow map, for the prediction of trajectories in N-body systems. In particular, we compare the accuracy of SympNets against standard multilayer perceptrons, both inside and outside the range of training data. We analyze our findings using a novel view on the topology of SympNets. Additionally, we also compare SympNets against classical symplectic numerical integrators. While the benefits of symplectic integrators for Hamiltonian systems are well understood, this is not the case for SympNets.

1 INTRODUCTION

The N-body problem is a surprisingly hard problem to solve, even though it is written down quite easily. The difficulty stems from its chaotic nature and from the fact that no analytical solution exists for $N \geq 3$.

Resolving a chaotic 3-body problem may require very small time steps in a classical algorithm and may therefore cost a lot of computing time. It may be the main bottleneck in larger N-body

simulations to resolve a close encounter of three bodies. A reasonable question to ask is: Can machine learning or more precisely neural networks in the form of a surrogate model help? A first paper trying to answer this question was published in 2020 [1].

The neural networks presented in [1] show extraordinary precision even for long integration times. However, these neural networks also show some shortcomings of which the most fundamental is the inability to predict for a longer time than a fixed time t_{end} defined through the data. We analyze how integrating fundamental physical properties in the neural networks can be helpful in creating surrogate models for the N-body problem. Possible benefits of neural networks with embedded physics include: better extrapolation outside the training data, more regularized predictions inside the range of training data and improved optimization behavior due to the reduced search space.

2 HAMILTONIAN SYSTEMS

The system of ODEs describing an autonomous Hamiltonian system can be written as:

$$\begin{pmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{q}} \end{pmatrix} = -J \nabla H(\boldsymbol{p}, \boldsymbol{q}), \quad J = \begin{pmatrix} 0 & I_d \\ -I_d & 0 \end{pmatrix}, \quad \boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^d. \tag{1}$$

It is called autonomous since the Hamiltonian $H : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ does not depend on time t. In some Hamiltonian systems the Hamiltonian is separable, meaning:

$$H(\boldsymbol{p}, \boldsymbol{q}) = T(\boldsymbol{p}) + U(\boldsymbol{q}), \tag{2}$$

where $T : \mathbb{R}^d \to \mathbb{R}$ is referred to as the kinetic energy and $U : \mathbb{R}^d \to \mathbb{R}$ as the potential energy. Hence, $H(\boldsymbol{p}, \boldsymbol{q})$ is the total energy of the system.

Autonomous Hamiltonian systems have two important properties, the first being conservation of the total energy along trajectories:

$$\frac{\mathrm{d}}{\mathrm{d}t} H(\boldsymbol{p}(t), \boldsymbol{q}(t)) = 0. \tag{3}$$

Secondly, the flow map $\varphi_h : \boldsymbol{x}(t) \mapsto \boldsymbol{x}(t+h)$ of the Hamiltonian system is symplectic, meaning:

$$\left(\frac{\partial \varphi_h}{\partial \boldsymbol{x}} \right)^T J \, \frac{\partial \varphi_h}{\partial \boldsymbol{x}} = J, \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{q} \end{pmatrix} \in \mathbb{R}^{2d}. \tag{4}$$

This property is characteristic for Hamiltonian systems, i.e., every system with a symplectic flow map is at least locally a Hamiltonian system [2, Theorem 2.6].

The Hamiltonian of the gravitational N-body problem can be given as

$$H(\boldsymbol{p}, \boldsymbol{q}) = \underbrace{\frac{1}{2} \boldsymbol{p}^T M^{-1} \boldsymbol{p}}_{T(\boldsymbol{p})} \underbrace{- \sum_{i=1}^{N} \sum_{j=1}^{i} G \frac{m_i m_j}{||\boldsymbol{q}_j - \boldsymbol{q}_i||}}_{-U(\boldsymbol{q})}, \tag{5}$$

where $d = 3N$, G the gravitational constant and M the mass matrix. M consists of the masses m_i of the N bodies and reads in three-dimensional space:

$$M \coloneqq \operatorname{diag}(m_1, m_1, m_1, m_2, \cdots, m_N, m_N, m_N). \tag{6}$$

3 SYMPLECTIC NUMERICAL INTEGRATORS

Classically, if one wants to solve a Hamiltonian system one resorts to symplectic numerical integrators. Compared to standard numerical integrators like Runge-Kutta methods, symplectic numerical integrators preserve the symplectic structure. This means that the numerical flow map $\Phi_h : \boldsymbol{x}_n \mapsto \boldsymbol{x}_{n+1}$ defined by the numerical integrator is symplectic as well and therefore satisfies equation (4). This comes with good numerical stability properties. Next, we introduce a known symplectic numerical integrator that is explicit for separable Hamiltonians and can be important to understand SympNets. In Figure 1, we depict the type of numerical integrators - in the following mostly shortly called integrators - to be considered here: single-step methods ($n := n + 1$).

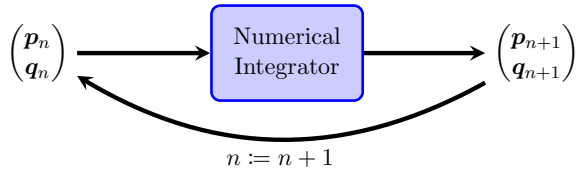

Figure 1: Classic iterative scheme to solve an ODE using a single-step numerical integrator.

According to [2], the symplectic Euler method for Hamiltonian systems, given $\boldsymbol{p}_n$ and $\boldsymbol{q}_n$, is defined as:

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n - h\nabla_q H(\boldsymbol{p}_{n+1}, \boldsymbol{q}_n), \tag{7}$$
$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n + h\nabla_p H(\boldsymbol{p}_{n+1}, \boldsymbol{q}_n). \tag{8}$$

For separable Hamiltonian systems this becomes an explicit method:

$$\begin{aligned}\boldsymbol{x}_n := \begin{pmatrix}\boldsymbol{p}_n\\ \boldsymbol{q}_n\end{pmatrix} \to \quad & \begin{pmatrix}\boldsymbol{p}_n - h\nabla U(\boldsymbol{q}_n)\\ \boldsymbol{q}_n\end{pmatrix} =: \begin{pmatrix}\boldsymbol{p}_{n+1}\\ \boldsymbol{q}_n\end{pmatrix} \to \\ \to \quad & \begin{pmatrix}\boldsymbol{p}_{n+1}\\ \boldsymbol{q}_n + h\nabla T(\boldsymbol{p}_{n+1})\end{pmatrix} =: \begin{pmatrix}\boldsymbol{p}_{n+1}\\ \boldsymbol{q}_{n+1}\end{pmatrix} =: \boldsymbol{x}_{n+1}.\end{aligned} \tag{9}$$

4 NEURAL NETWORKS FOR HAMILTONIAN SYSTEMS

The use of neural networks is very popular among stellar dynamicists. Whereas many specialized architectures for classical applications of machine learning exist already, specialized neural networks for Hamiltonian systems have only started to emerge during the last two years.

There are two ways of using neural networks to predict trajectories of Hamiltonian systems. Both are visualized in Figure 2. These two approaches are fundamentally different. Already the data to train these neural networks is different. The iterative neural network needs coordinates of a system as features and as labels the coordinates of the same system after a given time has passed. The actual time is irrelevant. It is only important that the time difference between features and labels is constant. Also, feature and label pairs do not need to lie on the same trajectory, they can stem from trajectories with different initial conditions. On the other hand, to train neural networks using the second approach one needs coordinates on trajectories at

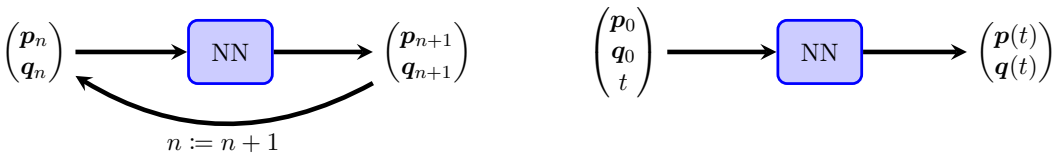

Figure 2: Iterative scheme to solve an ODE using a neural network (NN) compared to a non-iterative neural network requiring t as an input.

different times. Since time is an additional feature, the feature space has an additional dimension which needs to be covered during training, to prevent from moving outside the range of training data during inference.

Both approaches have benefits and drawbacks. While the iterative scheme suffers, just like a numerical integrator, from the accumulation of errors, the second approach always predicts the coordinates from the initial conditions. Hence, an error made at an earlier point in time does not influence the prediction at a later time. However, the second approach cannot be used to predict for very long times since it is bound to run out of the training data domain at some time. The iterative neural networks on the other hand can be used as long as the coordinates of the system stay within the training data. In the remainder of this paper we focus on iterative neural networks. The neural networks in [1] use the second approach.

4.1 Hamiltonian neural networks

The first structure-preserving neural network architecture for Hamiltonian systems was published by Greydanus et al. [3]. These neural networks, depicted in Figure 3, are named Hamiltonian neural networks (HNN).

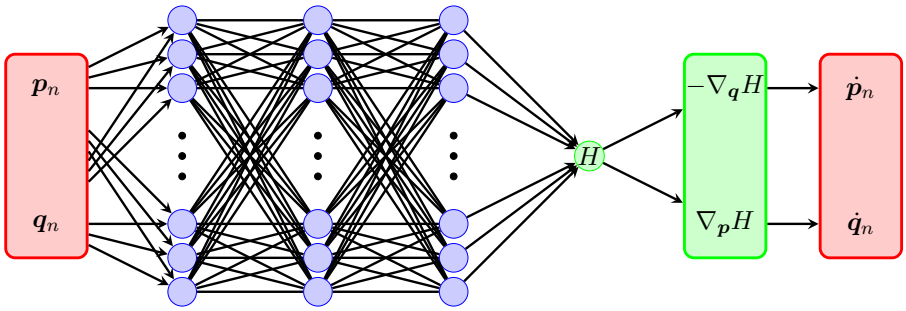

Figure 3: A Hamiltonian neural network as introduced in [3].

The key idea behind HNNs is to train the neural network to learn the Hamiltonian of a system without requiring data of the Hamiltonian itself. Instead, one can use automatic differentiation to calculate the derivatives of the neural network with respect to its inputs. According to equation (1) those derivatives correspond to the derivatives of the coordinates. Hence, one needs data that also includes the derivatives of the coordinates to evaluate a loss function. To train the neural network, the error is then backpropagated through the layer that calculates the derivatives and the rest of the neural network. During inference the derivatives of the neural

network can be used in combination with a numerical integrator to calculate updated positions and momenta of the system. Here any integrator with any step size can be used. Because the features and labels (coordinates and derivatives of coordinates) are taken at the same time, no inherent step size is fixed through the data. However, this changes with the next neural network architecture.

The second specialized neural network architecture for Hamiltonian systems was introduced in [4]. In [4], instead of training HNNs directly on the derivatives of the trained Hamiltonian, the numerical integrator is included in the neural network topology. The integrator is symplectic. In order to backpropagate the error through the integrator it is necessary to use an explicit integrator. Since explicit symplectic integrators only exist for separable Hamiltonian systems, the neural network is split into two detached neural networks. One predicts the kinetic energy and the second one the potential energy (Figure 4).

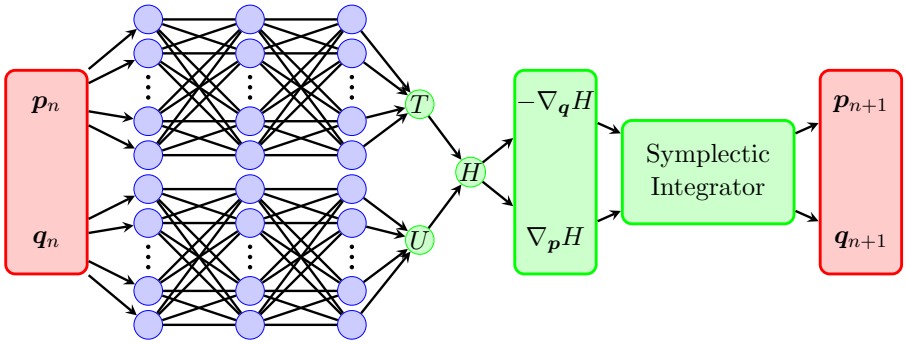

Figure 4: A Hamiltonian neural network as used in [4].

Because these neural networks directly learn the phase flow, they do not need data of the derivatives of the coordinates but time-stepped data instead. The requirement of more information of the system is traded against a few restrictions. These neural networks can only be used to learn separable systems. Furthermore, they are no longer free to use any numerical integrator and any time step once they are trained. The integrator is fixed in the neural network topology and the time step is fixed through the data used to train the neural network. One might think that during inference an integrator and time step can be used that are different from those used during training, since in the neural network a Hamiltonian is learned that can be extracted. However, it is already noted in [4] that the Hamiltonian in the neural network does not coincide with the real Hamiltonian of the system. Rather it is a Hamiltonian that is trained to achieve accurate predictions in combination with a specific integrator and time step. The predictions of these neural networks are even more accurate than predictions of the same integrator (as in the neural networks) using the same time step (as in the data) and the actual Hamiltonian of the system instead of the trained one. This can be explained by the trained Hamiltonian compensating for errors made by the integrator.

4.2 SympNets

The next type of neural networks tailor-made for Hamiltonian systems were published in [5] and called SympNets. At first glance they are completely different from the HNNs introduced so far. A SympNet never tries to learn the Hamiltonian of the system. Instead, it directly focuses on learning the symplectic flow map. In [5], different symplectic layers are introduced. Because a concatenation of symplectic maps is again a symplectic map, one can concatenate these layers indefinitely.

The standard representation of a SympNet can be seen in Figure 5. Every update of $\boldsymbol{p}$ or $\boldsymbol{q}$ with a trained gradient ∇V_i is called a module in [5]. Up and low modules are distinguished, indicating whether $\boldsymbol{p}$ or $\boldsymbol{q}$, respectively, is updated. In Figure 5, a SympNet with $2M$ alternating modules is displayed.

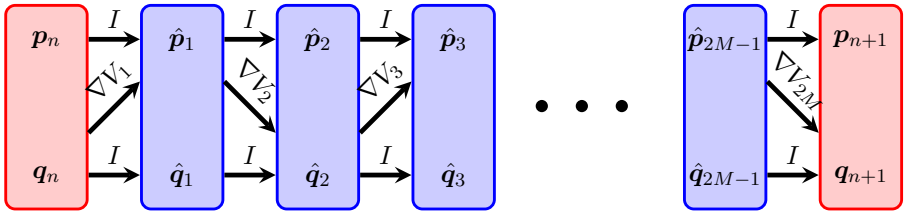

Figure 5: A standard SympNet from [5].

We now focus on the G-SympNets introduced in [5] and the so-called gradient modules. In a gradient module a trainable matrix $K \in \mathbb{R}^{n \times d}$ exists, which is similar to a weight matrix in a classical fully connected feed-forward neural network. In addition, a trainable bias $\boldsymbol{b} \in \mathbb{R}^n$ and scale factors $\boldsymbol{a} \in \mathbb{R}^n$ exist in a gradient module. An activation function σ 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 modules are defined as:

$$\mathcal{G}_{up}\begin{pmatrix}\boldsymbol{p}\\ \boldsymbol{q}\end{pmatrix} := \begin{bmatrix} I & \hat{\sigma}_{K,a,b} \\ 0 & I \end{bmatrix}\begin{pmatrix}\boldsymbol{p}\\ \boldsymbol{q}\end{pmatrix} = \begin{pmatrix}\boldsymbol{p} + K^T \mathrm{diag}(\boldsymbol{a})\sigma(K\boldsymbol{q} + \boldsymbol{b})\\ \boldsymbol{q}\end{pmatrix}, \tag{10}$$

$$\mathcal{G}_{low}\begin{pmatrix}\boldsymbol{p}\\ \boldsymbol{q}\end{pmatrix} := \begin{bmatrix} I & 0 \\ \hat{\sigma}_{\tilde{K},\tilde{a},\tilde{b}} & I \end{bmatrix}\begin{pmatrix}\boldsymbol{p}\\ \boldsymbol{q}\end{pmatrix} = \begin{pmatrix}\boldsymbol{p}\\ \boldsymbol{q} + \tilde{K}^T \mathrm{diag}(\tilde{\boldsymbol{a}})\sigma(\tilde{K}\boldsymbol{p} + \tilde{\boldsymbol{b}})\end{pmatrix}. \tag{11}$$

The name of this module comes from the fact that $\hat{\sigma}_{K,a,b}$ can approximate any gradient of a function $V : \mathbb{R}^d \to \mathbb{R}$ (so any ∇V).

Because of this property of the gradient modules, a universal approximation theorem for the G-SympNets can be proven [5, Theorem 5]. The theorem states that one can use G-SympNets to approximate the flow map of a Hamiltonian system up to any desired accuracy. There is no restriction to separable Hamiltonian systems. This is the same kind of universal approximation theorem that has been proven for the standard fully connected feed-forward neural networks

(multi-layer perceptron, shortly: MLP). Note that an MLP can achieve this universal approximation with only one hidden layer. To prove the universal approximation of SympNets, neither the depth nor the width of the neural networks can be fixed.

Additional research on the topic of neural networks for non-separable Hamiltonian systems and a different neural network topology are presented in [6]. The neural networks in [6] are more in line with HHNs and use an augmented phase space to be also applicable to non-separable Hamiltonian systems.

We now focus on SympNets since for the N-body problem we know the Hamiltonian, so there is no use in learning it. However, we want to see if it is possible to predict the trajectories of the bodies faster or with higher accuracy than by using numerical integrators. Furthermore, we study the effects of including the symplectic structure in the neural networks compared to a simple MLP.

5 Connection between SympNets and HNNs

Before presenting the numerical experiments for the 3-body problem, we take a closer look at the connection between SympNets and HNNs. This helps to understand some of our results, when comparing SympNets with the physics-unaware MLPs. Even though SympNets have been presented so far as completely different from HNNs, we are able to find a connection between both neural network topologies.

To understand this connection we rename the SympNet updates for $\boldsymbol{p}$ as ∇U and those for $\boldsymbol{q}$ as ∇T, instead of ∇V for both. If we also combine one up and one low module together and count these as a single combined module, we get a SympNet as visualized in Figure 6.

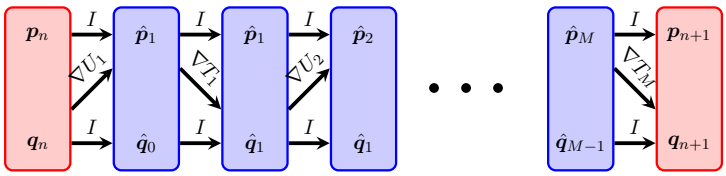

Figure 6: A SympNet with renamed updates.

So far, this is still very reminiscent of the visualization in Figure 5, but now we take a closer look at one combined module. Since we focus on G-SympNets, let us assume that the updates of $\boldsymbol{p}$ and $\boldsymbol{q}$ are done through gradient modules. From the definition of gradient modules one can easily calculate the functions $U(\boldsymbol{q})$ and $T(\boldsymbol{p})$, of which the gradients are used in the updates:

$$\nabla U(\boldsymbol{q}) = \nabla_{\boldsymbol{q}} \boldsymbol{a}^T \Sigma(K\boldsymbol{q} + \boldsymbol{b}) = K^T \mathrm{diag}(\boldsymbol{a}) \sigma(K\boldsymbol{q} + \boldsymbol{b}), \tag{12}$$

$$\nabla T(\boldsymbol{q}) = \nabla_{\boldsymbol{p}} \tilde{\boldsymbol{a}}^T \Sigma(\tilde{K}\boldsymbol{p} + \tilde{\boldsymbol{b}}) = \tilde{K}^T \mathrm{diag}(\tilde{\boldsymbol{a}}) \sigma(\tilde{K}\boldsymbol{p} + \tilde{\boldsymbol{b}}). \tag{13}$$

Here, Σ is the anti-derivative of the activation function σ. Using this we can write the updates in a combined module in a new way, as displayed in Figure 7.

In Figure 7, one recognizes the resemblance of the update structure to the symplectic Euler method. Indeed, this combined module performs one symplectic Euler step (with step size h)

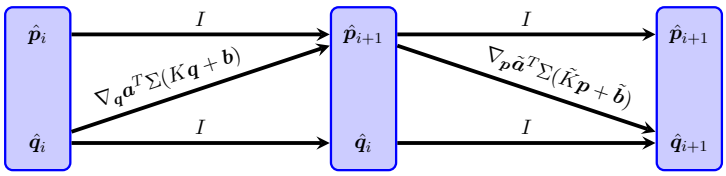

Figure 7: One symplectic gradient module of a SympNet.

for the learned Hamiltonian H given by the following formula:

$$H(\boldsymbol{p}, \boldsymbol{q}) = \underbrace{\frac{1}{h}\tilde{\boldsymbol{a}}^T \Sigma(\tilde{K}\boldsymbol{p} + \tilde{\boldsymbol{b}})}_{T(\boldsymbol{p})} \underbrace{- \frac{1}{h}\boldsymbol{a}^T \Sigma(K\boldsymbol{q} + \boldsymbol{b})}_{-U(\boldsymbol{q})}. \tag{14}$$

Additionally, the structure of the functions U and T is very specific. It is the structure of a simple feed-forward neural network with one hidden layer. The neural network for U receives $\boldsymbol{q}$ as input, has weight matrix K, bias vector $\boldsymbol{b}$, activation function Σ and weight matrix $\boldsymbol{a}^T$ in the output layer. The same holds for T, with different trainable parameters. Therefore, every two gradient modules of a SympNet also learn a hidden Hamiltonian inside, which can be expressed as the sum of two MLPs with one hidden layer each. This is visualized in Figure 8.

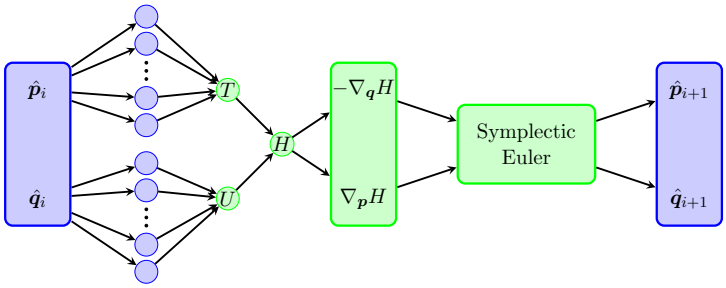

Figure 8: One symplectic gradient module of a SympNet visualized similar to a Hamiltonian neural network.

This graph looks similar to the one in Figure 4. In fact, one combined gradient module is an HNN with one hidden layer for U and T and a specific choice of a symplectic integrator, the symplectic Euler method. Therefore, a SympNet with $2M$ modules is equivalent to a concatenation of M independent specific HNNs. Each HNN uses one hidden layer and the symplectic Euler method. The only difference is that the gradients of the Hamiltonian in an HNN are calculated using automatic differentiation, while the gradients in a SympNet are pre-calculated. Using this equivalence the final visualization of a SympNet is given in Figure 9.

By concatenating HNNs, SympNets are able to approximate the symplectic flow map of all Hamiltonian systems even non-separable ones. A single HNN only is appropriate for separable Hamiltonian systems.

This analysis can also be performed for the activation and linear modules introduced in [5]. However, the analysis becomes less elegant and the notation more cluttered then. One would

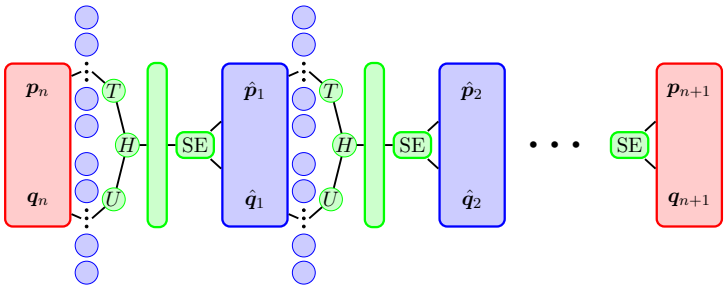

Figure 9: A SympNet using the new point of view (with SE meaning Symplectic Euler).

need to split the linear modules into the different sublayers since there is more than one triangular update in one linear module in general. As a consequence, the number of learned hidden Hamiltonians is larger but the individual Hamiltonians are simpler and no longer universal approximators.

6 NUMERICAL EXPERIMENTS

With numerical experiments we study the benefits of using the physics-aware SympNets in comparison with completely physics-unaware MLPs. Also, since we are in a scenario where we know the Hamiltonian of the system and therefore can simply use a numerical integrator, we compare SympNets against integrators like the symplectic Euler method.

6.1 Data

The data used in this research is created using the Brutus N-body solver [7]. Brutus uses an arbitrary-precision library combined with the Bulirsch-Stoer method to integrate Newton's equations of motion. The eventual N-body trajectories calculated using Brutus are converged solutions with a predefined tolerance. Because of the extreme precision and accuracy of Brutus the trajectories are numerically as close as technically possible to the ground truth.

The neural networks are trained on two different datasets. The first dataset is practically the same as the one used in [5]. It shares all the simplifications but is created using Brutus instead of a Runge-Kutta method. This dataset consists of 5000 trajectories of three bodies. The bodies are set to have equal masses 1, and $G = 1$ [8]. Every value given is dimensionless. All bodies are in the x, y-plane on a circle around the origin with the radius sampled randomly between 0.9 and 1.2. On this circle, the three bodies are 120° apart from each other. The initial velocities are calculated such that all bodies would perfectly orbit the origin on said circle. Finally, each of the velocities is multiplied by an independent random factor between 0.8 and 1.2. The system is then integrated up to $t_{\text{end}} = 10$ ($t_{\text{end}} = 5$ in [5]).

The second dataset consists of 1000 trajectories of two bodies in a solar system. The first body is the central star and the second a planet orbiting it. Both bodies move in the x, y-plane. The star has a mass of $1.7861 \cdot 10^{29}$ kg while the planet has a mass of $8.2058 \cdot 10^{24}$ kg. The semi-major axis and eccentricity are taken from a normal distribution centered around $1.7264 \cdot 10^{9}$ m and $6.22 \cdot 10^{-3}$, with a standard deviation of $7.4799 \cdot 10^{7}$ m and 0.1, respectively. All other orbital elements are set to 0. The system is integrated up to $t_{\text{end}} = 2 \cdot 10^{6}$ s. These parameter values

are the values of the star and the inner most planet in the Trappist-1 system [9]. However, in a simplified (projected on the x, y-plane) and slightly randomized form.

6.2 SympNets versus MLPs

To compare the SympNets against the physics-unaware MLPs we trained them on exactly the same time-stepped data generated from our first dataset. We cut the data at a final time $t_{\text{end}} = 4.5$. The motivation to do this is that for all 5000 trajectories the three bodies start close to a stable configuration in which they would periodically orbit their center of mass. Therefore, in the beginning all trajectories are rather similar.

We investigate how well the neural networks are able to predict the trajectories inside the trained time frame and also outside. To be able to compare these different neural network architectures we choose their width and depth such that their compute times for one trajectory are the same and such that both neural networks are able to achieve reasonably low losses. This leads to MLPs with 5 layers and 75 neurons per layer and SympNets with 20 modules and 50 units per module. In total this implies ∼70000 trainable parameters for the MLP and 8000 for the SympNet. The results of this study can be seen in Figure 10.

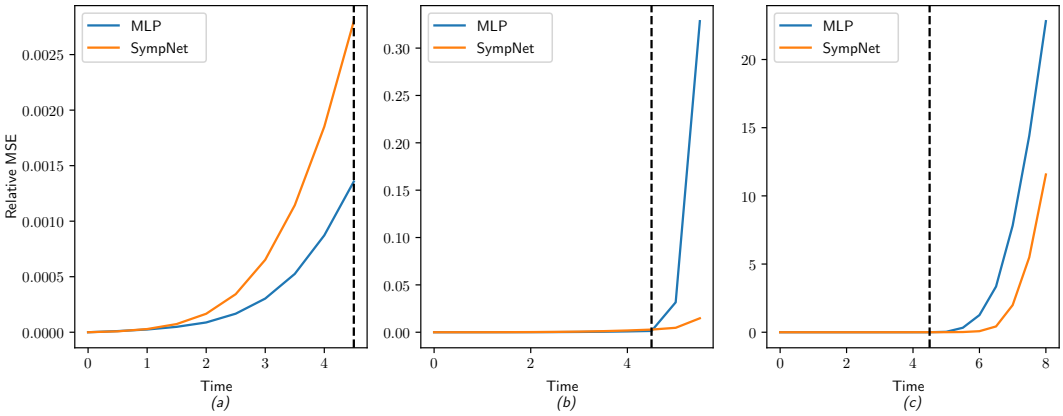

Figure 10: Relative mean square error (MSE) over the full dataset for an MLP and a SympNet trained on data until $t_{\text{end}} = 4.5$ (dashed vertical line): *(a)* inside training data, *(b)* outside training data, *(c)* far outside training data.

The clear exponential growth in the left most panel, that is still fully inside the data, can be explained by the iterative nature of this approach. Since the input to the neural networks already has an error after one time step, the prediction of the next step is even less accurate. The same behavior can be seen in numerical integrators and is expected. It is interesting to note that the MLP is more accurate inside the data than the SympNet. This means that even though the SympNet has the correct mathematical structure embedded in its topology, the MLP is able to approximate the flow map more accurately. Here, the high number of parameters outperforms the mathematical structure.

However, if we step outside the range of training data, even just one time step, the predicted trajectories by the SympNet are more accurate, and after another time step even an order of

magnitude more accurate. Hence, the SympNets are able to generalize (or extrapolate), whereas physics-unaware MLPs are not. One possible explanation for the ability to generalize are the learned hidden Hamiltonians inside the SympNet.

6.3 SympNets versus the symplectic Euler method

Since we know the Hamiltonian of the N-body problem as long as we know the masses of the bodies, the question arises why one would even use a neural network, instead of a numerical integrator. Therefore, we compare the performance of the symplectic Euler method with a SympNet on a rather simple 2-body system. We choose a single planet orbiting a star because it can be solved analytically and it results in a periodic solution, giving us the possibility to perform long-term predictions without chaos interfering.

We train a SympNet on data with a rather large time step of $h = 2 \cdot 10^4$ s, to see whether there is a break-even point in the size of the time step beyond which using a neural network becomes more efficient. A basic integration scheme like the symplectic Euler method will always be faster to calculate a single step than a large neural network, at least for a Hamiltonian that is easy to compute. However, a neural network might be able to predict over larger time steps than the symplectic Euler method and therefore outperform the integrator that needs many time steps.

For the symplectic Euler method we choose a time step of $h = 7 \cdot 10^3$ s to match the computing time of the neural network for a full trajectory until $t_{\text{end}} = 2 \cdot 10^6$ s. The results of this comparison can be seen in Figure 11 for one exemplary orbit of the dataset. The positions predicted by the

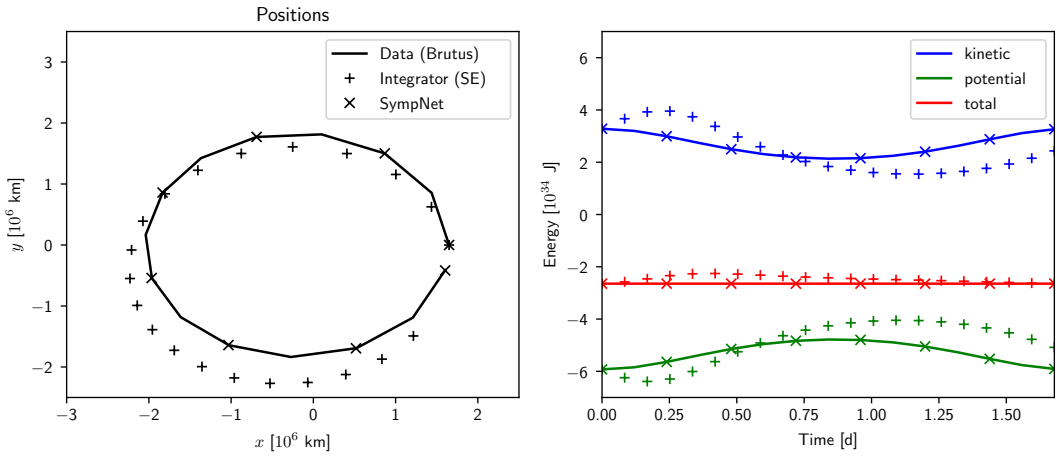

Figure 11: Trajectories of a planet orbiting a heavy central body, predicted by the symplectic Euler method (SE) and a SympNet. The step size for the Euler method is chosen such that it matches the speed of the neural network.

symplectic Euler method drift away from the true solution because the time step is too large for precise predictions. In contrast, the SympNet can perform the even larger time steps very accurately. The same behavior can be seen in the energy, even though a symplectic numerical integrator is used.

7 CONCLUSION

The symplectic structure in neural networks has other benefits than that in symplectic numerical integrators. While symplectic integrators ensure long-term energy conservation and stability, symplectic neural networks are not more stable than physics-unaware neural networks inside the range of training data. However, the SympNets perform much better outside the training data than the physics-unaware MLPs. This can be explained by the hidden Hamiltonians they learn from the data. All in all, while we are able to find better extrapolation outside the training data, we neither observe more regularized predictions inside the range of training data nor improved optimization behavior due to the reduced search space.

Furthermore, it can be concluded that even if the Hamiltonian of a system is known, it might still be beneficial to use a neural network to integrate trajectories. In the case of large time steps, a numerical integrator has difficulty to correctly calculate coordinates and therefore drifts away from the correct solution over time. A neural network can learn to predict new coordinates over long time steps. Therefore, if the state of a system after a long time is of interest, a neural network can deliver accurate predictions faster than a numerical integrator.

REFERENCES

[1] P.G. Breen, C.N. Foley, T. Boekholt and S. Portegies Zwart, Newton versus the machine: solving the chaotic three-body problem using deep neural networks, *Monthly Notices of the Royal Astronomical Society*, Vol. **494**, pp. 2465-2470, 2020.

[2] E. Hairer, C. Lubich and G. Wanner, Geometric Numerical Integration, Springer, 2006.

[3] S. Greydanus, M. Dzamba and J. Yosinski, Hamiltonian neural networks, *Advances in Neural Information Processing Systems*, Vol. **32**, NeurIPS, 2019.

[4] Z. Chen, J. Zhang, M. Arjovsky and L. Bottou, Symplectic recurrent neural networks, *8th International Conference on Learning Representations*, ICLR, 2020.

[5] P. Jin, Z. Zhang, A. Zhu, Y. Tang and G.E. Karniadakis, SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems, *Neural Networks*, Vol. **132**, pp. 166-179, 2020.

[6] S. Xiong, Y. Tong, X. He, S. Yang, C. Yang and B. Zhu, Nonseparable symplectic neural networks, *9th International Conference on Learning Representations*, ICLR, 2021.

[7] T. Boekholt and S. Portegies Zwart, On the reliability of N-body simulations, *Computational Astrophysics and Cosmology*, Vol. **2**, 2014.

[8] D.C. Heggie and R.D. Mathieu, Standardised units and time scales, *The Use of Supercomputers in Stellar Dynamics, Lecture Notes in Physics*, Vol. **267** , pp. 233–235, Springer, 1986.

[9] M. Gillon, E. Jehin, S. Lederer, et al., Temperate Earth-sized planets transiting a nearby ultracool dwarf star, *Nature*, Vol. **533**, pp. 221–224, 2016.