# Improving Simulations with Symmetry Control Neural Networks

**Anonymous authors**
Paper under double-blind review

## Abstract

The dynamics of physical systems is often constrained to lower dimensional sub-spaces due to the presence of conserved quantities. Here we propose a method to learn and exploit such symmetry constraints building upon Hamiltonian Neural Networks. By enforcing cyclic coordinates with appropriate loss functions, we find that we can achieve improved accuracy on simple classical dynamics tasks. By fitting analytic formulae to the latent variables in our network we recover that our networks are utilizing conserved quantities such as (angular) momentum.

## 1 Introduction

For accurate simulations, building in a bias to deep neural networks has been a key mechanism to achieve extra-ordinary performance. An example for such a bias is to learn a motion which is conserving energy as performed in the context of Hamiltonian Neural Networks (HNNs) (Greydanus et al., 2019). More generally speaking, possible dynamics are constrained due to symmetries of the system and they take place on a sub-space of available phase space (e.g. surfaces of constant energy and total angular momentum). Whereas energy has been built into the functional biasing of neural networks, further symmetries are at this moment enforced by hand which requires domain knowledge of the system. Here we describe a method on how to automatically search for and use such symmetries in neural networks used for simulations. In addition this method turns out to reveal – at this stage for simple systems – analytic expressions for the conserved quantities. Our method is based on enforcing the network to perform canonical coordinate transformations in a first network $T_\psi$ which enforces some of the latent coordinates to become constant (cyclic coordinates in the language of Hamiltonian dynamics). From these new coordinates the network then predicts the Hamiltonian underlying the dynamics of this system with a second network $H_\phi$. The structure of our networks – to be described more accurately later on – can be found in Figure 1. We demonstrate in our experiments that multiple conserved quantities can indeed be automatically identified with this method, we find improvements in the predictions of our dynamics in comparison with HNNs, and we can find analytic expressions for the conserved quantities.

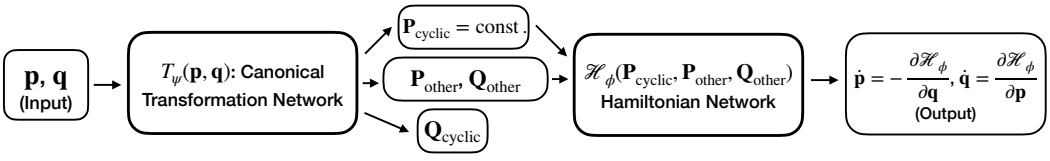


Figure 1: Structure of symmetry control networks: The network $T_\psi$ transforms the input coordinates of phase space $(\mathbf{p}, \mathbf{q})$ to canonical coordinates where some coordinates are forced to be cyclic. These coordinates are used as input to the Hamiltonian network $H_\phi$. The output, the time derivatives of our initial coordinates, is calculated from the Hamiltonian using auto-differentiation.

## 2 THEORY

When motion is taking place on a sub-space of phase space, one can reduce the number of coordinates describing the system. Every independent conserved quantity reduces the number of coordinates needed to describe a system until all coordinates are constrained (integrable systems). Even though we do not know apriori the conserved coordinates, Hamiltonian mechanics provides general conditions for such coordinates. In particular the time derivative of any function on phase space, described by positions $\mathbf{q}$ and momenta $\mathbf{p}$ of our $N$ particles in $d$ spatial dimensions, is given by

$$\frac{dg(\mathbf{q},\mathbf{p})}{dt} = \sum_{i=1}^{N\cdot d} \frac{\partial g}{\partial q_i}\frac{dq_i}{dt} + \frac{\partial g}{\partial p_i}\frac{dp_i}{dt} = \{g, \mathcal{H}\} \ . \tag{1}$$

In the last step we have used the Poisson brackets and Hamilton's equations which are given as:

$$\{f,g\} := \sum_{i=1}^{N\cdot d} \frac{\partial f}{\partial q_i}\frac{\partial g}{\partial p_i} - \frac{\partial f}{\partial p_i}\frac{\partial g}{\partial q_i} \,, \qquad \frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{d\mathbf{p}} = \{\mathbf{q}, \mathcal{H}\} \,, \qquad \frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{d\mathbf{q}} = \{\mathbf{p}, \mathcal{H}\} \,. \tag{2}$$

Here we are interested in enforcing coordinate transformations $T$ which enforce that at least one of the new momenta $\mathbf{P}$ is conserved

$$0 = \dot{P}_i = -\frac{\partial \mathcal{H}}{\partial Q_i} = \{P_i, \mathcal{H}\} \,. \tag{3}$$

We are interested in a particular type of coordinate transformations, namely canonical transformations which are which are transformations that leave the structure of the Hamiltonian equations (2) and in particular the Poisson bracket unchanged (allowing for the evaluation with respect to the input variables):

$$\begin{aligned} &T : (\mathbf{q},\mathbf{p}) \mapsto (\mathbf{Q}(\mathbf{q},\mathbf{p}), \mathbf{P}(\mathbf{q},\mathbf{p})) \,, \\ &\{f,g\}_{\mathbf{p},\mathbf{q}} = \{f,g\}_{\mathbf{P},\mathbf{Q}}, \ \mathcal{H}(\mathbf{p},\mathbf{q}) = \tilde{\mathcal{H}}(\mathbf{P}(\mathbf{p},\mathbf{q}), \mathbf{Q}(\mathbf{p},\mathbf{q})) \ . \end{aligned} \tag{4}$$

We enforce this conservation by adding an appropriate loss term corresponding to the condition in (3). When such a constraint is enforced, the Hamiltonian does not depend on the associated $Q_i$, i.e. it depends on fewer degrees of freedom and the motion in phase space is restricted to a lower dimensional manifold. Put differently, the cyclic coordinates provide via constraints of the type (3) additional restrictions on the allowed Hamiltonian function which we learn with our symmetry control neural networks.

These conditions lead to two additional loss components in addition to the standard HNN-loss:

1. The first loss ensures that our Hamiltonian satisfies Hamiltonian equations (2), which we can ensure as follows:

$$\mathcal{L}_{\text{HNN}} = \sum_{i=1}^{N\cdot d} \left\| \frac{\partial \mathcal{H}_\phi(\mathbf{P},\mathbf{Q})}{\partial p_i} - \frac{dq_i}{dt} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\phi(\mathbf{P},\mathbf{Q})}{\partial q_i} + \frac{dp_i}{dt} \right\|_2 \ . \tag{5}$$

   The time derivatives are provided by the data and the derivatives of the Hamiltonian with respect to the input variables can be obtained using auto-differentiation. This is the same loss as introduced in Greydanus et al. (2019).

2. To ensure that our transformation $T_\psi$ are of the type we are interested in (cf. Eq. (4)), i.e. our new coordinates fulfill the Poisson algebra, we enforce the following loss:

$$\mathcal{L}_{\text{Poisson}} = \sum_{i,j=1}^{N\cdot d} \|\{Q_i, P_j\} - \delta_{ij}\|_2 + \sum_{i,j>i}^{N\cdot d} \|\{P_i, P_j\}\|_2 + \|\{Q_i, Q_j\}\|_2 \ , \tag{6}$$

   where in some practical applications we only enforce this loss on $n$ cyclic coordinate pairs. The first part of this loss ensures that a vanishing solution is not allowed.

3. Hamilton's equations have still to be satisfied with respect to the new coordintaes. For the cyclic coordinates we have enforced by our architecture that $\mathcal{H}_\phi$ is independent of $Q_i$. To ensure that $P_i$ is actually conserved, we require the following additional loss:

$$\begin{aligned}\mathcal{L}^{(n)}_{\text{HQP}} =& \sum_{i=1}^{n} \left\| \frac{dP_i}{dt} \right\|_2 + \left\| \frac{dQ_i}{dt} - \frac{\partial \mathcal{H}_\phi(\mathbf{P},\mathbf{Q})}{\partial P_i} \right\|_2 \\ &+\beta \sum_{i=n+1}^{N\cdot d} \left\| \frac{dP_i}{dt} + \frac{\partial \mathcal{H}_\phi(\mathbf{P},\mathbf{Q})}{\partial Q_i} \right\|_2 + \left\| \frac{dQ_i}{dt} - \frac{\partial \mathcal{H}_\phi(\mathbf{P},\mathbf{Q})}{\partial P_i} \right\|_2 , \end{aligned} \tag{7}$$

where $n$ denotes the number of cyclic variables we are imposing and $\beta$ denotes a hyperparameter. For our networks we only constrain the cyclic coordinates and use $\beta = 0$. The time derivatives can be calculated using either expressions in (1).

Our total loss is a weighted sum of these three components:

$$\mathcal{L} = \mathcal{L}_{\text{HNN}} + \alpha_1 \mathcal{L}_{\text{Poisson}} + \alpha_2 \mathcal{L}^{(n)}_{\text{HQP}} , \tag{8}$$

where the weights $\alpha_i$ are tuned.

For integrating the solutions in time from our respective symmetry control network we use a fourth order Runge-Kutta integrator as in Greydanus et al. (2019) which unlike symplectic integrators allows for a comparison with neural network approaches directly predicting the dynamics of a system.[1]

## 3 Experiments

Our experiments[2] are designed with the following goals in mind:

1. **SCNN:** We want to compare the performance of symmetry control neural networks with HNNs and baseline neural networks which directly predict $(\dot{\mathbf{q}}, \dot{\mathbf{p}})$. We want to take advantage of unknown underlying symmetries while we do not have any domain knowledge of our system. The main hyperparameter is the number of conserved quantities.
2. **SCNN-constraint:** We explore whether imposing domain knowledge about symmetries improves the performance. This is motivated by the fact that we often know about the existence of certain conserved quantities such as (angular) momentum.

We test these architectures on two different datasets: the gravitational two-body problem and an $n$-body system with spring-like interactions. The spring-like interaction contained between 3 and 5 particles each interacting with the force $\mathbf{F^{ij}} = -k\ \left(\mathbf{x}^i - \mathbf{x}^j\right)$. The gravitational force is described by $\mathbf{F^{ij}} = -k\ \frac{(\mathbf{x}^i - \mathbf{x}^j)}{|\mathbf{x}^i - \mathbf{x}^j|^3}$.

To understand the effect of the number of conserved quantities and the loss prefactors $\alpha_i$, we performed a parameter scan for the number of conserved quantities and these prefactors. To compare the precision of the networks we computed the particle trajectories of 100 different system initializations and compared them to the ground truth over 100 timesteps. In addition, we check the conservation of energy and angular momentum for the gravitational 2-body problem. Our networks are trained on datapoints corresponding to the first 50 timesteps which allows us to compare the generalization of our networks on unseen regions of phase-space. The results are summarized in Figure 3. Overall, these results show that the SCNN learns a much more stable and accurate Hamiltonian and we find that the SCNN-constraint networks are not able to outperform the SCNN networks. The baseline network is omitted from these plots as its performance is significantly worse. The HNN network has the same complexity as our $\mathcal{H}_\phi$ network with two dense layers and the HNN_L network

[1] We find that the main numerical inaccuracy in the prediction arises from inaccuracies of the Hamiltonian $\mathcal{H}_\phi$ rather than the choice of this integrator when comparing it with standard symplectic integrators (Rein & Liu, 2012).

[2] Details on hyperparameters and training can be found in the appendix.

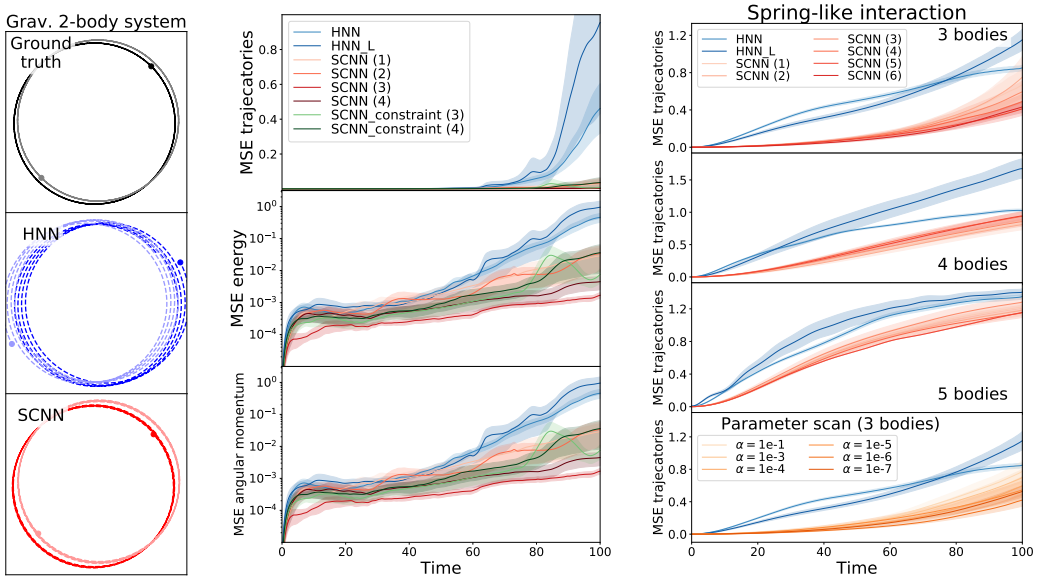


Figure 2: Comparison between two HNN networks as described in the main text and SCNN networks with different numbers of conserved quantities. **Left:** Sample trajectories of the gravitational two-body problem. **Middle:** Comparison of the mean-squared deviation of the trajectories, the conservation of energy and angular momentum for the 2-body problem. **Right:** Comparison of the mean-squared deviations for various spring-like interactions. The bottom plot shows the well-behaved dependence on the loss prefactor $\alpha = \alpha_i$. For all evaluations of the trajectories we show the mean and single standard deviation over 10 different initializations evaluated on 100 trajectories. $\alpha_1 = \alpha_2 = 10^{-n}$ where $n$ is the number of constraint conserved quantities.

corresponds to a complexity of five dense layers, i.e. the combined complexity of our $T_\psi$ and $\mathcal{H}_\phi$ network. The loss prefactor dependence is such that the more conserved quantities we use, the smaller the prefactors have to be which is due to the increase in loss terms contributing.

### Search for conserved quantities

The conserved quantities of our SCNN networks can be analysed by fitting a low-dimensional polynomial ansatz to the respective network predictions. This reveals that our SCNN finds the angular momentum and the total momentum as conserved quantities in the gravitational two-body system:

$$P_{c_1} = -4.2p_{x_1} - 4.2p_{x_2} - 1.3p_{y_1} - 1.3p_{y_2}\ ,\ P_{c_2} = -0.9p_{x_1} - 0.9p_{x_2} - 3.2p_{y_1} - 3.2p_{y_2}\ ,$$
$$L = -1.1q_{x_1}p_{y_1} + 0.9q_{x_1}p_{y_2} + 0.9q_{x_2}p_{y_1} - 1.0q_{x_2}p_{y_2} + 1.0q_{y_1}p_{x_1} - 0.9q_{y_1}p_{x_2} - 0.9q_{y_2}p_{x_1} + 1.0q_{y_2}p_{x_2}\ . \tag{9}$$

For more sophisticated conserved quantities (i.e. non-polynomial conserved quantities) different ansätze seem necessary (some of which are pursued in related work (Sahoo et al., 2018; Cranmer et al., 2019; Wetzel et al., 2020).

## 4 Outlook and related work

Our SCNNs naturally connect with work on inferring dynamics with neural networks such as (Battaglia et al., 2016) in the same way as HNNs. Natural extensions of our current work can include the application on graph neural network based approaches (Sanchez-Gonzalez et al., 2019) and in flows (Toth et al., 2019). With respect to applications, we particularly look forward to applying our new approach for automatically inferring and using the symmetries in astrophysical and molecular dynamics settings.

## References

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Miles D Cranmer, Rui Xu, Peter Battaglia, and Shirley Ho. Learning symbolic physics with graph networks. *arXiv preprint arXiv:1909.05862*, 2019.

Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *CoRR*, abs/1906.01563, 2019. URL `http://arxiv.org/abs/1906.01563`.

H. Rein and S. F. Liu. REBOUND: an open-source multi-purpose N-body code for collisional dynamics. *AAP*, 537:A128, January 2012. doi: 10.1051/0004-6361/201118085.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4442–4450, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/sahoo18a.html`.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. *arXiv preprint arXiv:1909.13789*, 2019.

Sebastian J. Wetzel, Roger G. Melko, Joseph Scott, Maysum Panju, and Vijay Ganesh. Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks, 2020.

## A Hyperparameters

We have performed some hyperparameter searches for our symmetry control networks on which we give an overview here.

By varying the hidden layer size between 50-300, we find that there is a minimum size of layer size 200 to find convergence of our networks. We have varied the number of hidden layers up to 5 hidden layers. Two hidden layers are already sufficient (tested up to 5 hidden layers), hence we restricted ourselves on them. Coarsely speaking, we find that the precise architecture is less relevant in our current experiments.

More relevant are the pre-factors in the loss. Depending on the choice, we can either force our networks for better performance on the predictions or obtaining the conserved quantities. We have used $\alpha_i$ in the range $[10^{-7}, 1]$. For this trade-off, we have optimized the experiments in this paper on the particle trajectories.

We find that pytorch's standard orthogonal initialization provides the best results out of the standard initializations. We have not observed a large random seed dependence.

As datasets we use the nearly circular orbits constructed by Greydanus et al. (2019), but give the whole system a boost in a random direction sampled from $\mathcal{N}(0, 0.1)^2$. For the spring-like interaction, we initialize the first $n-1$ particles with $(\mathbf{q}, \mathbf{p})$ from $\mathcal{U}([-1.5, 1.5])^4$. For the $n$-th particle, we use $(\mathbf{q}, \mathbf{p}) = (\sum_i \mathbf{q}_i, \sum_i \mathbf{p}_i)$, which ensures that the center of mass is at rest.