Symmetry Control Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

This paper continues the quest for designing the optimal physics bias for neural networks predicting the dynamics of systems when the underlying dynamics shall be inferred from the data directly. The description of physical systems is greatly simplified when the underlying symmetries of the system are taken into account. In classical systems described via Hamiltonian dynamics this is achieved by using appropriate coordinates, so-called cyclic coordinates, which reveal conserved quantities directly. Without changing the Hamiltonian, these coordinates can be obtained via canonical transformations. We show that such coordinates can be searched for automatically with appropriate loss functions which naturally arise from Hamiltonian dynamics. As a proof of principle, we test our method on standard classical physics systems where our network identifies the conserved quantities in an unsupervised way and find improved performance on predicting the dynamics of the system compared to networks biasing just to the Hamiltonian. Effectively, these new coordinates guarantee that motion takes place on symmetry orbits in phase space, i.e. appropriate lower dimensional subspaces of phase space. By fitting analytic formulae we recover that our networks are utilising conserved quantities such as (angular) momentum.

1 Introduction

Building in a bias to neural networks has been a key mechanism to achieve extra-ordinary performance in tasks such as classification. A standard example is to utilise translation invariance in convolutional neural networks Krizhevsky et al. (2012) and by now building in equivariance to other symmetries such as rotational symmetries has proven to be very successful (e.g. Cohen & Welling (2016)).

Possible motions are constrained due to symmetries of the system. In technical terms, motion takes place on a subspace of phase space. Energy conversation – related to invariance under time translation – has been utilised in the context of Hamiltonian Neural Networks (HNNs) Greydanus et al. (2019) where the energy functional, i.e. the Hamiltonian is inferred from data. This approach has seen large improvements in predicting the dynamics over baseline neural networks which simply try to predict the change of phase-space coordinates in time. Here we extend this approach by learning and incorporating additional constraints due to further symmetries of the system.

Coarsely speaking, finding symmetries corresponds to finding good coordinates. In classical mechanics this is achieved by performing canonical transformations and identifying cyclic coordinates which reveal conserved quantities. The aim of this paper is to demonstrate that multiple conserved quantities can indeed be automatically found in this way which has not been demonstrated beforehand.

Similar in spirit to learning the Hamiltonian, we formulate loss functions which enforce a representation in terms of cyclic coordinates and use them as the input for our Hamiltonian, differing from previous flow-based approaches searching for these coordinates (Bondesan & Lamacraft, 2019; Li et al., 2020). We experimentally find as a proof of principle that this mechanism identifies the underlying conserved quantities such as angular momentum, momentum, and the splitting into decoupled subsystems.

We demonstrate significant improvement in the predictions of the dynamics and the conservation of conserved quantities with our network setup compared to HNNs and baseline neural network approaches.

2 Theory

We briefly describe the standard techniques in Hamiltonian mechanics which our network utilises.[1] We consider a classical system with N particles in d spatial dimensions. Such a system can be described by the variables $(\mathbf{q}, \mathbf{p})$, where $\mathbf{q} = (q_1, ..., q_{N \cdot d})$ are typically the positions for each dimension of the objects and $\mathbf{p} = (p_1, ..., p_{N \cdot d})$ are the corresponding momenta. This is the input to our network and we are interested in predicting the time-evolution of this system, i.e. $(\mathbf{q}, \mathbf{p})$ at later time steps.

This pair $(\mathbf{q}, \mathbf{p})$ is an element of phase space in which every point corresponds to a state the system can take. The time evolution is governed by the Hamiltonian $\mathcal{H}(\mathbf{q}, \mathbf{p})$[2] and the associated Hamiltonian equations:

$$\frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{d\mathbf{p}} = \{\mathbf{q}, \mathcal{H}\}\,, \qquad \frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{d\mathbf{q}} = \{\mathbf{p}, \mathcal{H}\}\,, \tag{1}$$

where $\{\bullet, \bullet\}$ are the Poisson bracket. They are defined as

$$\{f, g\} := \sum_{i=1}^{N \cdot d} \frac{\partial f}{\partial q_i} \frac{\partial g}{\partial p_i} - \frac{\partial f}{\partial p_i} \frac{\partial g}{\partial q_i}\,. \tag{2}$$

The Poisson bracket does not only arise for the time evolution of the canonical coordinates $(\mathbf{q}, \mathbf{p})$ but also for any function of these coordinates $g(\mathbf{q}, \mathbf{p})$ which does not explicitly depend on time:

$$\frac{dg(\mathbf{q}, \mathbf{p})}{dt} = \sum_{i=1}^{N \cdot d} \frac{\partial g}{\partial q_i} \frac{dq_i}{dt} - \frac{\partial g}{\partial p_i} \frac{dp_i}{dt} = \{g, \mathcal{H}\}\ , \tag{3}$$

where we have used the Hamiltonian equations (1) in the last step. From this expression we see that the Poisson bracket of a conserved quantity with the Hamiltonian $\mathcal{H}(\mathbf{q}, \mathbf{p})$ vanishes and that the Hamiltonian itself is a conserved quantity.

The physics of this system is invariant under diffeomorphic coordinate transformations on the canonical coordinates and we will use coordinate transformations to reveal the constants of motions and hence the symmetries of the system. We use a particular type of diffeomorphic transformations, namely *canonical transformations* which are transformations that leave the structure of the Hamiltonian equations (1) and in particular the Poisson bracket unchanged:

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

We are interested in finding transformations T such that at least one coordinate satisfies

$$0 = \dot{P}_i = -\frac{\partial \mathcal{H}}{\partial Q_i} = \{P_i, \mathcal{H}\}\,. \tag{5}$$

Such a coordinate P_i is conserved in the system and 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.

Symmetry Control Neural Networks: In a first step we search for such *cyclic coordinates* with a network T_ψ and use them as input for our Hamiltonian $\mathcal{H}_\phi$. The structure of our two trainable networks is shown in Figure 1. In order to find cyclic coordinates and to identify a Hamiltonian, our loss function contains several components:

[1] We refer the reader for more details to standard textbooks such as Landau & Lifshitz (1982).

[2] We focus on time-independent Hamiltonians for simplicity.

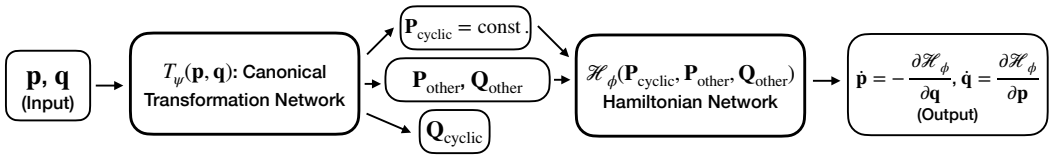

Figure 1: Structure of symmetry control networks: The network T_ψ transforms the input coordinates of phase space $(\mathbf{p}, \mathbf{q})$ to canonical coordinates where some coordinates are forced to be cyclic. These coordinates are used as input to the Hamiltonian network H_ϕ. The output, the time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

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

$$\mathcal{L}_{\text{HNN}} = \sum_{i=1}^{N \cdot d} \left\| \frac{\partial \mathcal{H}_\phi(\mathbf{P}, \mathbf{Q})}{\partial p_i} - \frac{dq_i}{dt} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\phi(\mathbf{P}, \mathbf{Q})}{\partial q_i} + \frac{dp_i}{dt} \right\|_2 . \tag{6}$$

   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. Our new coordinates have to fullfil the Poisson algebra which is enforced with the following loss:

$$\mathcal{L}_{\text{Poisson}} = \sum_{i,j=1}^{N \cdot d} \left\| \{Q_i, P_j\} - \delta_{ij} \right\|_2 + \sum_{i,j>i}^{N \cdot d} \left\| \{P_i, P_j\} \right\|_2 + \left\| \{Q_i, Q_j\} \right\|_2 , \tag{7}$$

   where in some practical applications we only enforce this loss on n cyclic coordinate pairs.

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

$$\begin{aligned} \mathcal{L}^{(n)}_{\text{HQP}} = & \sum_{i=1}^{n} \left\| \frac{dP_i}{dt} \right\|_2 + \left\| \frac{dQ_i}{dt} - \frac{\partial \mathcal{H}_\phi(\mathbf{P}, \mathbf{Q})}{\partial P_i} \right\|_2 \\ & + \beta \sum_{i=n+1}^{N \cdot d} \left\| \frac{dP_i}{dt} + \frac{\partial \mathcal{H}_\phi(\mathbf{P}, \mathbf{Q})}{\partial Q_i} \right\|_2 + \left\| \frac{dQ_i}{dt} - \frac{\partial \mathcal{H}_\phi(\mathbf{P}, \mathbf{Q})}{\partial P_i} \right\|_2 , \end{aligned} \tag{8}$$

   where n denotes the number of cyclic variables we are imposing and β denotes a hyperparameter for our experiments. The time derivatives can be calculated using either expressions in (3).

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

$$\mathcal{L} = \mathcal{L}_{\text{HNN}} + \alpha_1 \mathcal{L}_{\text{Poisson}} + \alpha_2 \mathcal{L}^{(n)}_{\text{HQP}} , \tag{9}$$

where the weights α_i are tuned.

As the number of conserved quantities in a system is generally unknown, we use an iterative process starting with one cyclic coordinate and subsequently search for more and more conserved quantities.

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).

3 Experiments

In our experiments we compare our approach with the performance of HNNs and baseline architectures. We are interested to see whether our learned symmetry bias can be beneficial to predict the time-evolution of systems and whether we are successful in finding the conserved quantities in our setup. We also obtain analytic expressions for the learned quantities to check for agreement with known conserved quantities. For this comparison we use several systems already used in Greydanus et al. (2019) but with slightly different parameter ranges as described below.

3.1 Physical Systems and Objective of Experiments

We consider the **two-body problem** in two dimensions which is governed by the following Hamiltonian

$$\mathcal{H} = \frac{p_{x1}^2}{2m_1} + \frac{p_{y1}^2}{2m_1} + \frac{p_{x2}^2}{2m_2} + \frac{p_{y2}^2}{2m_2} - \frac{g}{\|\mathbf{q}_1 - \mathbf{q}_2\|_2} \ . \tag{10}$$

To simplify the problem we set $m_1 = m_2 = g = 1$. We generate datasets with near circular orbits where the whole system has a small velocity in an arbitrary direction to avoid accidental symmetries (see Appendix A for more details). Besides the Hamiltonian itself, conserved quantities are the total momentum in x and y direction, and the angular momentum:

$$\begin{aligned} P_x = p_{x1} + p_{x2} \ , \qquad P_y = p_{y1} + p_{y2} \ , \\ L = (p_{x1} - p_{x2})\,(q_{y1} - q_{y2}) - (p_{y1} - p_{y2})\,(q_{x1} - q_{x2})\,. \end{aligned} \tag{11}$$

The second system is the **3-body system** which is governed by the following Hamiltonian:

$$\mathcal{H} = \sum_{i=1}^{3} \frac{p_{xi}^2}{2m_i} + \frac{p_{yi}^2}{2m_i} - \frac{1}{2} \sum_{i=1, j\neq i}^{3} \frac{g}{\|\mathbf{q}_i - \mathbf{q}_j\|_2} \ , \tag{12}$$

where we take a univeral mass and coupling $m_i = g = 1$. As in the 2-body example, we consider systems which have a small velocity in an arbitrary direction (cf. Appendix A). Again, the overall momentum in both directions and the angular momentum are conserved quantities:

$$\begin{aligned} P_x = p_{x1} + p_{x2} + p_{x3} \ , \qquad P_y = p_{y1} + p_{y2} + p_{y3} \ , \\ L = (2q_{x1} - q_{x2} - q_{x3})\,p_{y1} - (2q_{y1} - q_{y2} - q_{y3})\,p_{x1} + \text{even permutations} \ . \end{aligned} \tag{13}$$

The third system which we are interested in is the **n-dimensional coupled oscillator** which is described by the following Hamiltonian

$$\mathcal{H} = \sum_{i=1}^{n} \frac{p_i^2}{2m} + \sum_{i,j=1}^{n} q_i A_{ij} q_j \ , \tag{14}$$

where we use $m = 1/2$ and the coupling matrix A_{ij} is a symmetric $n \times n-$matrix with positive eigenvalues. As this matrix is not diagonal, the potential energy of each oscillator in these coordinates is influenced by the position of the other oscillators. By diagonalization the system can be described by decoupled harmonic oscillators. The conserved quantities are the energies of these subsystems:

$$E_i = \tilde{p}_i^2 + \lambda_i \tilde{q}_i^2 \ , \tag{15}$$

where $\tilde{p}$ and $\tilde{q}$ are the canonical coordinates associated to the respective eigenvalue λ_i of the coupling matrix A_{ij}. Note that these systems include spring-like interactions.

For these systems, we now present the results from several experiments where we have tested our symmetry controlled network structure described in Section 2. Our objectives were as follows:

- Does our setup converge and utilise known conserved quantities in the canonical coordinates? Are these known conserved quantities actually conserved during for the predicted trajectory?
- Can we improve the performance for predicting the future trajectory by utilising these conserved quantities sing our architecture in comparison to HNNs?

3.2 2-BODY PROBLEM: RESULTS

We have trained our networks with data from 800 trajectories with 50 datapoints each and checked the performance on 200 different trajectories with 50 datapoints (cf. Appendix A for more details). For our experiments we have used Adam as optimizer with a learning rate of 10^{-3} and no weight decay as our optimiser. The implementation of our experiments is submitted alongside with this paper. More details on the hyperparameter choices can be found in Appendix B. We compare the following architectures which all converge to a reasonable accuracy and our large training set ensures that overfitting does not pose a problem here:

- The **baseline architecture** has two hidden dense layers with 200 hidden units and tanh activation. The output is directly $(\dot{\mathbf{q}}, \dot{\mathbf{p}})$.
- **HNN:** We use the original architecture with two hidden dense layers, both networks using tanh activations and 200 hidden units.
- **Model 1:** we use a transformation network architecture with two hidden dense layers (tanh activation) and a Hamiltonian neural network with two hidden dense layers as above. We enforce four cyclic coordinates. For this network we use a loss with $\alpha_{1,2} = 0.001$ and $\beta = 0$.
- **Model 2:** We utilise a transformation network which contains a single dense layer transformation. Two output variables are forced to be cyclic. Consequently our Hamiltonian network takes six input quantities and the rest of the Hamiltonian network is unchanged. For this network we use a loss with $\alpha_{1,2} = 0.01$, $\beta = 0$, and the Poisson loss is only enforced on cyclic coordinates.
- **Model 3:** Here we fix both conserved momenta as in Equation (11) and require two additional cyclic coordinate. For the trainable coordinates there are still two hidden dense layers with 200 units. For this network we use a loss with $\alpha_{1,2} = 0.001$, $\beta = 0$, and the Poisson loss is only enforced on cyclic coordinates.
- In **Model 4** we fix the conserved momenta and do not search for additional conserved quantities beyond the Hamiltonian which is calculated with our standard two hidden dense layers from the input $(\mathbf{q}, \mathbf{p})$. Our loss is $\alpha_1 = \beta = 0$ and in the $\mathcal{L}^n_{\text{HQP}}$−loss ($\alpha_2 = 1.0$) we ignore the second term in the sum which is constraining $\dot{Q}_i$. This ensures that the Poisson bracket of the conserved momenta with the Hamiltonian vanishes.
- **Model 5** is the same as Model 4 but we also fix the angular momentum and ensure that the respective Poisson bracket with the Hamiltonian vanishes.

For all networks we analyse the time evolution predicted from our trained network by determining the trajectory, the deviation from the ground truth and the evolution of the conserved quantities. A sample for the predicted trajectories is shown for each network in Figure 2. To quantify the accuracy of the respective networks we show the mean-squared deviation from the ground truth trajectories which we obtain over 50 different trajectories on our trained networks. We see an order of magnitude improvement with our symmetry control networks compared to the HNN and the baseline networks. Finally the bottom line of plots in Figure 2 shows the deviation of conserved quantities predicted from our networks in comparison to the underlying conserved quantity. Our results, which are obtained by averaging again over 50 trajectories, show that symmetry control networks provide an order of magnitude improvement on the conservation of several quantities. We notice a better performance of our symmetry control networks when we fix several known conserved quantities.

Model 1 reveals the following three conserved quantities:

$$\begin{aligned} P_{c1} &= -\,4.21\; p_{x1} - 4.21\; p_{x2} - 1.26\; p_{y1} - 1.29\; p_{y2}\;, \\ P_{c2} &= -\,0.93\; p_{x1} - 0.92\; p_{x2} - 3.23\; p_{y1} - 3.22\; p_{y2}\;, \\ L &= -\,1.07\; q_{x1}p_{y1} + 0.88\; q_{x1}p_{y2} + 0.93\; q_{x2}p_{y1} - 1.03\; q_{x2}p_{y2} \\ &\quad + 1.01\; q_{y1}p_{x1} - 0.89\; q_{y1}p_{x2} - 0.92\; q_{y2}p_{x1} + 0.99\; q_{y2}p_{x2}\;. \end{aligned} \tag{16}$$

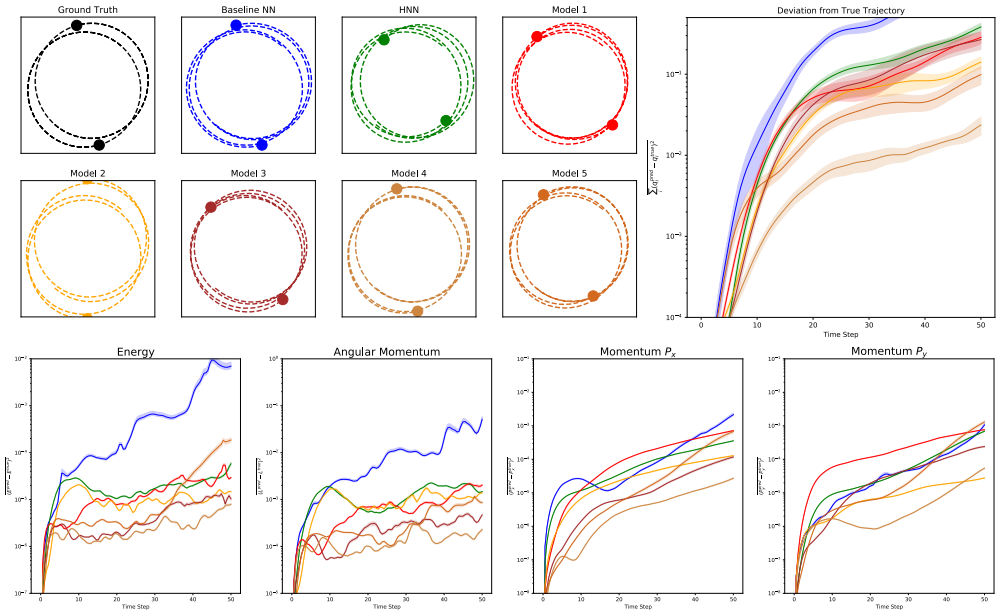

Figure 2: **Top left:** Sample trajectories for the ground truth, the baseline neural network, a HNN, and our symmetry control networks in the 2-body problem. The colour coding is such that the same colour is used throughout all plots in this figure. **Top right:** Shows the deviation of predicted from the true trajectory. The solid line denotes the mean deviation over 50 trajectories and the one sigma band is shown shaded around it. **Bottom:** We show mean squared deviation from conserved quantities in the 2-body problem where again we show the mean deviation over 50 trajectories and and the associated one sigma band.

The first two are clear linear combinations of the conserved momenta in Equation (11) and the third one is corresponding to angular momentum. To find these analytic fits we have simply used a polynomial ansatz and checked that there are no significant changes when allowing for higher powers in the analytic ansatz.[3] Sub-leading terms are omitted where the largest ones of these terms arise with a factor smaller than 0.1.

3.3 Three-body problem

Apart from adapting the input and output dimensions appropriately, we keep the baseline and HNN architecture unchanged to the two-body problem. For our symmetry control networks we report results for three architectures which are variants of the symmetry control networks for the two-body problem:

- **Model 1** is the same architecture as Model 1 in the two-body case but we search for 6 cyclic coordinates.
- **Model 2:** Here we fix both conserved momenta as in Equation (13) and require one additional cyclic coordinate. For the trainable coordinates there are still two hidden dense layers with 200 units. For this network we use a loss with $\alpha_{1,2} = 0.001$, $\beta = 0$, and the Poisson loss is only enforced on cyclic coordinates.
- **Model 3:** is the same architecture as Model 5 in the two-body case where we fix all three conserved quantities as in Equation (13).

We train our networks starting again at near circular orbits (cf. Appendix A). However, after an initial circular motion for around 10 time steps the motion changes drastically when two

[3]Throughout our experiments we sometimes find that the iterative procedure of identifying a few conserved quantities first works even better.

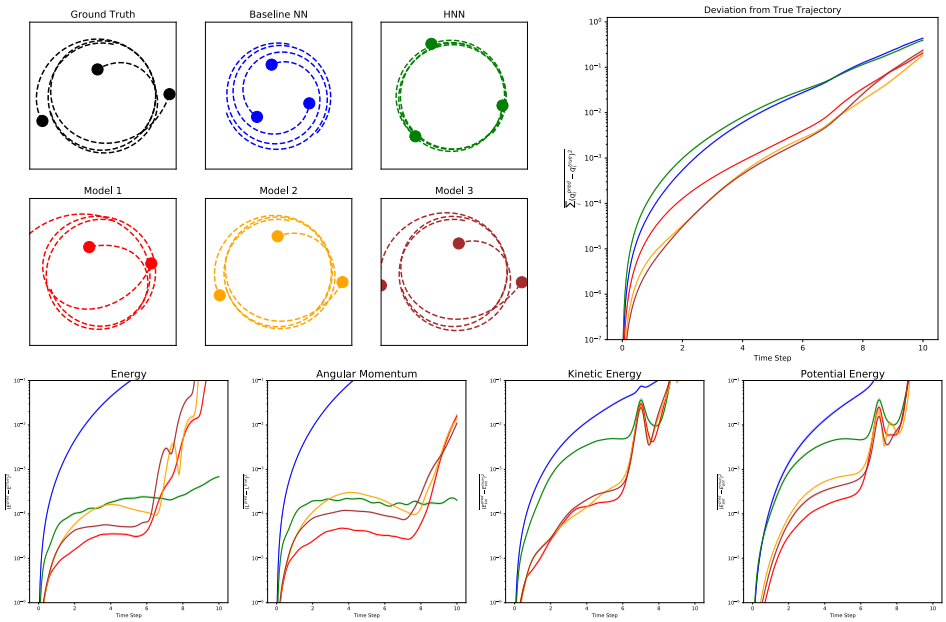

Figure 3: **Top left:** Sample trajectories for the ground truth, the baseline neural network, a HNN, and our symmetry control networks in the 3-body problem. The colour coding is such that the same colour is used throughout all plots in this figure. **Top right:** Shows the deviation of predicted from the true trajectory. The solid line denotes the mean deviation over 50 trajectories and the one sigma band is shown shaded around it. **Bottom:** We show mean squared deviation from conserved quantities in the 3-body problem where again we show the mean deviation over 50 trajectories and and the associated one sigma band.

particles come close to each other. In our experiments, we are not able to capture these dynamics and we do not aim to predict the dynamics after this initial drastic deviation. We observe in our experiments that our symmetry control networks are capable to predict this initial drastic change correctly. This is in contrast to the HNN and baseline networks whose predictions for the trajectories are worse than for the symmetry control networks and do not capture this change. We show an example of the trajectories in Figure 3, alongside with the mean deviation of the trajectory, and the conservation of angular momentum, energy and the prediction of kinetic and potential energy.

3.4 Coupled harmonic oscillator

Again we keep the HNN architecture and the architectures in comparison to our previous experiments unchanged apart from adapting the input and output dimensions. As these experiments do not pose a large challenge for either HNNs and our symmetry control neural networks, we would like to highlight one additional point regarding fitting analytic expressions for the conserved quantities and the Hamiltonian. For instance, in the case of $n = 2$, we find that the conserved quantities show explicitly the decoupling of the two sub-systems in our symmetry control networks. We find for the conserved quantities the underlying energies of the subsystem, e.g.:

$$\begin{aligned} P_1 &= 0.67\ q_1^2 - 3.6\ q_1 q_2 + 8.23\ q_2^2 + 0.46\ p_1^2 - 1.78\ p_1 p_2 + 4.2\ p_2^2 + \dots, \\ P_2 &= -4.53\ q_1^2 - 1.74\ q_1 q_2 - 0.9\ q_2^2 - 4.5\ p_1^2 - 1.88\ p_1 p_2 - 0.57\ p_2^2 + \dots\ , \end{aligned} \tag{17}$$

where the largest of the neglected terms arises with a factor smaller than 0.05. The entire Hamiltonian is also fitted very accurately as

$$H = 1.02\ q_1^2 - 0.41\ q_1 q_2 + 1.89\ q_2^2 + 0.97\ p_1^2 + 0.99\ p_2^2 + \dots\ , \tag{18}$$

where the largest term which is neglected has a factor smaller than 0.05. For these experiments we have used a symmetry control network as in Model 1 with two cyclic coordinates.

We observe that varying the relative loss weights $\alpha_i = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ does not affect the results significantly. Details regarding the data we have used for training and testing can be found in Appendix A.

When finding such good agreement with the underlying analytic formulae, it becomes easier to predict the particle trajectory directly by using this expression for the Hamiltonian, i.e. essentially to follow the procedure used to produce the ground truth trajectories.

4 Related Work

There are several related directions where symmetries have been learned and utilised with neural networks. Also the use of physics bias not just via the Hamiltonian framework is playing a large role in related works. We briefly comment on several of them:

- The baseline approach to predict dynamics has been studied in many other approaches previously (e.g. Grzeszczuk et al. (1998) for early work on this topic).
- An alternative to describing dynamics via Hamiltonian is via Lagrangian. The equivalent approach to HNNs in terms of learning Lagrangians is discussed in (Cranmer et al., 2020; Gupta et al., 2019; Lutter et al., 2019).
- Canonical transformations to cyclic coordinates have been utilised in the context of flow-networks. Bondesan & Lamacraft (2019) aim at finding the canonical transformation to cyclic coordinates but they do not study the improved predictions using those coordinates and their loss uses finite difference as opposed to an auto-differentiation loss we utilise here. Li et al. (2020) requires all coordinates to be cyclic (i.e. the system is integrable) which is not the case in our approach.
- Predicting the coordinates $(\mathbf{q}, \mathbf{p})$ from images has been pursued in (Greydanus et al., 2019; Toth et al., 2019). Here this would add an additional network before our input with the target output $(\mathbf{q}, \mathbf{p})$ as was explicitly demonstrated to work in (Toth et al., 2019).
- Very accurate prediction of dynamics can be performed using interaction networks (Battaglia et al., 2016; Watters et al., 2017). Biasing the local interactions via the Hamiltonian in graph networks has been discussed in Sanchez-Gonzalez et al. (2019). It would be very interesting to apply our approach in this framework to analyse the performance at larger systems.
- Fitting analytic formulae for the network output on physical systems has been pursued in (Cranmer et al., 2019; Wetzel et al., 2020).

5 Discussion

One great challenge in physics is the search for conserved quantities and underlying symmetries. Our architecture provides an approach on how to find these symmetries using neural networks automatically without prior knowledge of them. We find interesting performance on simple toy systems where the symmetries are well known. We are able to identify these symmetries and by using them we can improve on performance with Hamiltonian neural networks.

It will be extremely interesting to extend on this proof of concept analysis by applying it to state of the art predictive networks such as in (Sanchez-Gonzalez et al., 2019) which as of now only utilise the Hamiltonian and not all of the symmetries.

A second line of investigation is in the realm of physics. An example for an area of future application is in integrable systems which for classical systems is satisfied if all coordinates are cyclic. Determining whether a system is integrable is generally speaking unknown and our method provides an automated route to search for the required structures.

We hope to report on progress in these directions at future conferences.

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.

Roberto Bondesan and Austen Lamacraft. Learning symmetries of classical integrable systems. *arXiv preprint arXiv:1906.04645*, 2019.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

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`.

Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 9–20, 1998.

Jayesh K. Gupta, Kunal Menda, Zachary Manchester, and Mykel J. Kochenderfer. A general framework for structured learning of mechanical systems, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

L.D. Landau and E.M. Lifshitz. *Mechanics: Volume 1*. Number v. 1. Elsevier Science, 1982. ISBN 9780080503479. URL `https://books.google.de/books?id=bE-9tUH2J2wC`.

Shuo-Hui Li, Chen-Xiao Dong, Linfeng Zhang, and Lei Wang. Neural canonical transformation with symplectic flows. *Physical Review X*, 10(2):021020, 2020.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

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.

Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Advances in neural information processing systems*, pp. 4539–4547, 2017.

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 Data

A.1 Two-body problem

As described in the main text we set both masses and the coupling to unity. We sample 1000 trajectories with 50 data points each over a time span of 20 which corresponds of at least two third of a full circle. We uniformly sample the starting point for the first body $\mathbf{q}_1$ from the box $[0.5, 1.5]^2$. The starting position for the second body is at $\mathbf{q}_2 = -\mathbf{q}_1$.

We choose the velocity of both bodies in such a way that the bodies describe near circular orbits and we give the entire system an overall momentum $\mathbf{p}_{\text{com}} = (\epsilon_1, \epsilon_2)$

$$\begin{aligned} \mathbf{p}_1 &= (1+\epsilon)\ \mathbf{p}_{\text{circ}} + \mathbf{p}_{\text{com}}\ , \\ \mathbf{p}_2 &= -\mathbf{p}_1 + 2\ \mathbf{p}_{\text{com}}\ , \end{aligned} \tag{19}$$

where ϵ and $\epsilon_{1,2}$ are sampled from a normal distribution with $\sigma = 0.05$ and 0.1 respectively.

We take 80 percent of the data for training and use the rest for testing.

A.2 Three-body problem

The setup is relatively similar to the two-body problem as we consider three body near circular motions and give the overall system an overall momentum. We initialise the three particles on the corners of an equilateral triangle. The velocity of the first particle is given as follows

$$\mathbf{p}_1 = (1+\epsilon)\ \mathbf{p}_{\text{circ}} + \mathbf{p}_{\text{com}}\ , \tag{20}$$

where the remaining velocities are chosen accordingly as in the two-body problem and where the center of mass motion and the deviation from the circular motion are exactly generated as in the two-body example. We use 5000 trajectories with 20 data points for training and testing of our networks. The time span is 5 which corresponds to motion which is mostly circular. When using longer time spans, we find that convergence of the networks becomes problematic.

A.3 n-dimensional harmonic oscillator

The underlying Hamiltonian (14) is generated as follows. The coupling matrix is generated as the product of an orthogonal matrix S and a diagonal matrix D as $A_{ij} = SD^2S^T$. The entries of the diagonal matrix D are random entries sampled uniformly in the interval $[1 - \sqrt{10}]$ which are then rounded to one digit to ensure that the values are *sufficiently rational*, i.e. that the motion is periodic. For instance, if we choose an entry in D^2 equal to 2 the motion is not periodic whereas if we choose 1.96 the motion is periodic. The matrix S is obtained by generating random normal distributed entries (mean 0 and standard deviation 0) and we then orthonormalize the vectors.

The initial conditions $(\mathbf{q}(t=0), \mathbf{p}(t=0))$ are a $2n$-dimensional vector which we generate as follows. We generate a vector with random entries (uniform sample) in the interval $(0-2)$. We then normalize the vector to unit length and then re-scale it uniformly to a vector of length in the interval $(1, 10)$.

For our dataset we utilise 100 initial conditions and generate 500 points on the trajectory. These points are taken from a time span of 10 which corresponds to a time where multiple periodic motions are included. We split it into equal size training and test set.

B Hyperparameters

We have performed some hyper-parameter 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 100 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. 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.