Manuscript submitted to AIMS Journal

doi:10.3934/xx.xxxxxxx

PSEUDO-HAMILTONIAN SYSTEM IDENTIFICATION

Sigurd Holmsen[1], Sølve Eidnes*[2] and Signe Riemer-Sørensen[2]

[1]Norges Bank, 0107 Oslo, Norway

[2]Department of Mathematics and Cybernetics, SINTEF Digital, 0373 Oslo, Norway

Abstract. Identifying the underlying dynamics of physical systems can be challenging when only provided with observational data. In this work, we consider systems that can be modelled as first-order ordinary differential equations. By assuming a certain pseudo-Hamiltonian formulation, we are able to learn the analytic terms of internal dynamics even if the model is trained on data where the system is affected by unknown damping and external disturbances. In cases where it is difficult to find analytic terms for the disturbances, a hybrid model that uses a neural network to learn these can still accurately identify the dynamics of the system as if under ideal conditions. This makes the models applicable in some situations where other system identification models fail. Furthermore, we propose to use a fourth-order symmetric integration scheme in the loss function and avoid actual integration in the training, and demonstrate on varied examples how this leads to increased performance on noisy data.

1. **Introduction.**

Ever since the concept of describing physical systems by differential equations was introduced with the invention of calculus, humans have sought to identify the differential equations that most accurately describe a given system. The identification of the correct mathematical terms to include has historically been based on a combination of qualitative analysis, i.e. assumptions about the forces and physical laws that affect the system, and quantitative analysis, i.e. evaluation of data collected from experiments. As the capabilities of computers to store and handle large quantities of data increase, together with decreasing costs of sensors and computers, quantitative analysis can take an increasingly dominant role in system identification.

Two clear and important advantages of identifying analytic terms for a system, rather than learning a black-box model, stand out: For one, analytic terms can give insight into the process under consideration that may increase understanding and spur further developments. Secondly, accurate analytic models are less likely to suffer from the poor extrapolation abilities that make most machine learning models not applicable outside the range of the training data.

2020 *Mathematics Subject Classification.* Primary: 34A55, 37M10; Secondary: 37J99.

Key words and phrases. Physics-informed machine learning, hybrid modelling, dynamical systems, system identification.

This research was supported by the Research Council of Norway and the industry partners Elkem, Eramet Norway, Equinor, BP, Subsea7, Kongsberg Maritime, Aker Solutions and Veas, through the projects BigDataMine (project no. 309691), PRAI (Prediction of Riser-response by Artificial Intelligence) (project no. 308832) and PhysML: Structure-based machine learning for physical systems (project no. 338779).

*Corresponding author.

Recently, there has been a number of works on neural network models for dynamical systems. These aim at training a model to approximate the right-hand side of a first-order ordinary differential equation (ODE) system

$$\dot{x} = g(x,t), \quad x \in \mathbb{R}^d, \quad t \in \mathbb{R}. \tag{1}$$

Physics-informed machine learning is a field motivated by the idea that prior assumptions and knowledge should be imposed on the machine learning model so that it does not have to be relearned [46, 31, 56]. This is especially relevant for learning dynamical systems. Much of the literature has focused on Hamiltonian or Lagrangian formulations of (1), beginning with [23, 8, 10]. In [17, 20], it is demonstrated how a *pseudo-Hamiltonian* formulation can be utilized in the model structure to separate the internal dynamics and external forces acting on the system. This pseudo-Hamiltonian formulation is a generalization of the port-Hamiltonian formulation [54], which is again a generalization of the Hamiltonian formulation. The key innovation of these pseudo-Hamiltonian neural networks (PHNN) is their ability to learn a model for the full dynamical system (1) from data while simultaneously learning models for the external forces acting on the system. The external forces can be removed from the model after training, and thus a model trained under suboptimal conditions learns the properties of the system itself and hence can be used to model the system as if under optimal conditions.

This addresses a major limitation of system identification. Most frameworks struggle to learn a system when the data is sampled under suboptimal conditions, e.g. when there is some external disturbance affecting the dynamics; the frameworks cannot disentangle the disturbances from the system itself. The motivating idea behind this paper is to use the pseudo-Hamiltonian formulation to separate out external forces while identifying analytic terms for the internal dynamics. This can make system identification applicable for more complicated and realistic systems than those frequently studied in the literature.

Moreover, we tackle the problem of how to learn models of systems described by differential equations when we cannot assume to have data on derivatives. This has been extensively studied for neural network models [28, 39, 62], but less so for system identification models. We propose here to train on a numerical integration scheme without actually integrating so that we can even out the noise by including two data points in the discretization of (1).

This paper aims to advance the field of system identification toward a wider applicability to real-world problems through two main contributions:

- By assuming a pseudo-Hamiltonian formulation of the system, we propose models that can be used to learn conservation laws, damping and external forces simultaneously, where external forces can be learned by a neural network model if they are difficult to express analytically.
- We argue for using symmetric numerical integrators in the training of the model and demonstrate superior performance over existing methods, especially on noisy data.

The implementation of the PHSI models is done in Python and builds on the `phlearn` package introduced in [20]. Code to reproduce numerical experiments from the paper is published at https://github.com/SINTEF/PHSI.

2. Related work.

This paper stands on two broad shoulders: The first is the long history of and extensive literature on system identification, see e.g. [36], and especially recent developments that involve applying machine learning techniques and sparse representations, as made popular by the sparse identification of nonlinear dynamics (SINDy) framework [4]. The second consists of recent developments in data-driven models of dynamical systems informed of a certain structure and studies on numerical integrators in the training of these models, where most of the literature to date has been on neural network models.

The SINDy approach for system identification has been extensively studied and extended to a wide area of applications [45, 32]. These include areas where (pseudo-)Hamiltonian formulations are of interest, like control theory [30, 21] and partial differential equations [47, 48, 29], and hence our methodology could also be applicable here. The SINDy approach leverages L_1-regularized regression to prune the space of terms to include in the analytic expression for (1). This makes it less dependent on large amounts of data and more applicable for nonlinear systems than genetic algorithms [33] and symbolic regression [49]. However, there has been recent advances exploiting symbolic regression in different ways such as using filters for the derivatives and combining with a symbolic neural network [37], learning a graph neural network before applying symbolic regression [11], and AI Feynman [51], which provides a framework for system identification in a step-wise and recursive procedure that involves detecting structures in the system to reduce the search space. Although terms involving derivatives are not considered in [51], this could be incorporated to allow for discovering differential equations. Hence our proposed method could be integrated as a step in a framework similar to AI Feynman, where the identified pseudo-Hamiltonian structure would indicate which terms to include in the search space. In that case, it could be the next step after detecting conservation laws and neural network models of conserved quantities, e.g. as done in [35, 1, 40].

Hamiltonian neural networks (HNN) have received considerable attention since their introduction in [23], resulting in several extensions and generalizations [8, 22, 27]. Especially relevant for this paper are the generalizations of the Hamiltonian formulation allowing for damping, disturbances acting on the system, or interaction with other systems. In addition to those that use the term port-Hamiltonian [15, 17] or pseudo-Hamiltonian [20, 19], several papers consider a specific generalization of Hamiltonian systems that fall under the definition of pseudo-Hamiltonian systems applied in this paper, whether they aim to learn external forces [18] or not [60, 59, 58]. Our motivation for the pseudo-Hamiltonian formulation comes from the long history of modeling mechanical systems by studying the conservation of energy, momentum, mass, and other quantities [2, 38, 24]. Thus we consider systems with external forces of arbitrary form. This marks a slight distinction from the port-Hamiltonian formulation, which in addition involves some structure on the external forces, called interaction and control ports in that setting. The term port-Hamiltonian has its origin in control theory, where this formulation is viewed as a special realization of passive systems [55, 53, 54]. There are several recent works on system identification of the port-Hamiltonian formulation of passive systems specifically [3, 9, 42], which take a more classical approach to system identification than we do here, and assume the Hamiltonian to be quadratic.

Among the literature on HNN, several papers have proposed using symplectic integrators during training, since these will preserve the Hamiltonian within some time-independent bound when used for integration [8, 28, 13, 14]. It has been argued that using symplectic integrators in the training may allow for extracting qualitative properties e.g. through obtaining the Hamiltonian from the so-called inverse modified equation given perfect training [61, 62, 44]. However, convincing theoretical or numerical results on improved accuracy of the resulting models are lacking (see Appendix A). A study of symplectic integrators for learning pseudo-Hamiltonian systems is to our knowledge non-existing as of yet, but in this paper, we argue that symmetry is a more important property than symplecticity for integrators used for system learning, and is particularly well-suited for dealing with noisy data. We propose to use a fourth-order method, partly motivated by the convincing results from using higher-order methods for neural network models [14, 20, 43].

The works most closely related to ours are the recent papers on system identification of Hamiltonian systems, or a generalization of this, based on approaches similar to SINDy. When applicable, we compare our model to that of [16], which considers the identification of separable Hamiltonian systems. [34] also considers a port-Hamiltonian formulation, but only in the case where the external force is known a priori. Both these references apply a training algorithm that involves integrating the learned system as proposed for Neural ODEs [7]. By contrast, we use the integration scheme directly in the loss function, as is done for neural network models in [39, 28, 13, 20, 43].

3. Background.

In the experiments presented later in this paper we consider specific Hamiltonian and port-Hamiltonian formulations. However, to demonstrate the broader applicability of our approach, we first introduce a generalized framework, relevant for all systems that can be described by an ODE (1). Therefore we consider a more general formulation of energy-preserving systems than the canonical Hamiltonian formulation

$$\begin{pmatrix} \dot q \\ \dot p \end{pmatrix} = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix} \begin{pmatrix} \partial H/\partial q \\ \partial H/\partial p \end{pmatrix} \qquad q, p \in \mathbb{R}^n, H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \tag{2}$$

considered in e.g. [23, 16, 34]. Furthermore, for non-preserving systems we consider a more general formulation than the port-Hamiltonian formulation of e.g. [3, 15]. Following [20, 19], we call this formulation pseudo-Hamiltonian.

3.1. Hamiltonian systems.

The most general class of energy-preserving systems, encompassing canonical and non-canonical Hamiltonian systems, is given by

$$\dot x = S(x)\, \nabla H(x), \tag{3}$$

where x denotes the generalized coordinates, $S : \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$ is such that $S(x)$ is an antisymmetric matrix for any x, and $H : \mathbb{R}^d \to \mathbb{R}$ is the Hamiltonian. We continue to call it the Hamiltonian, although it can be any conserved quantity; given initial conditions x^0, it will be preserved at all subsequent times, i.e. $H(x(t)) = H(x^0)$. This property follows directly from the formulation (3) and the antisymmetry of S:

$$\dot H = \nabla H^T \dot x = \nabla H^T(x)\, S(x)\, \nabla H(x) = 0. \tag{4}$$

Even when H is not energy in the physical sense, (3) is still usually called an energy-preserving system [24]. Moreover, an ODE system may have several preserved quantities and can be written on the form (3) for different H's and corresponding non-unique S's [41]. If S is the symplectic matrix given in (2), (3) is a canonical Hamiltonian system; if $S(x)$ defines a Poisson bracket satisfying the Jacobi identity, it is a Poisson system [24]. In all experiments in this paper, we assume that S is constant and known.

3.2. **Pseudo-Hamiltonian formulation.** We desire a generalization of the Hamiltonian system formulation that allows for energy dissipation and external forces affecting the system. Thus we consider what we call *pseudo-Hamiltonian systems*:

$$\dot{x} = (S(x) - R(x))\,\nabla H(x) \,+\, F(x,t), \tag{5}$$

where $R : \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$ is such that $R(x)$ is a positive semi-definite matrix defining the dissipation for each coordinate of x, and $F : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ denotes the external forces. If we imposed certain conditions on $F(x,t)$, this would be equivalent with the port-Hamiltonian formulation from control theory [53, 54]. The formulation (5) is completely general and non-unique; e.g., we can set $S(x) = R(x) = 0$ and $F(x,t) = g(x,t)$ and recover (1). However, a model based on (5) sets up a framework for imposing assumptions about the system that will make the formulation less general, as we will demonstrate in Section 5.

3.3. **Numerical integration of ODEs.** When an analytic solution of (1) is not available, the evolution of the system from time t^n to time $t^{n+1} = t^n + \Delta t$ can be estimated by the use of a numerical integration scheme

$$\frac{x^{n+1} - x^n}{\Delta t} = \Psi_{\Delta t}(g, x^n, x^{n+1}, t^n), \tag{6}$$

where Ψ depends on the chosen integrator. If Ψ is explicitly given by x^n and t^n, we say that it is an explicit integrator; we have e.g. $\Psi_{\Delta t}(g, x^n, \cdot, t^n) = g(x^n, t^n)$ for the forward Euler method. Implicit integrators depend on x^{n+1} or intermediate steps, and thus a system of equations has to be solved by some root-finding algorithm when these are used for propagating non-linear ODEs. The main drawback of this class of integrators is generally considered to be their computational inefficacy compared to explicit integrators. However, a large class of implicit integrators is explicitly given by x^n and x^{n+1}; e.g.

$$\Psi_{\Delta t}(g, x^n, x^{n+1}, t^n) = g(\frac{x^n + x^{n+1}}{2}, t^n + \frac{\Delta t}{2}) \tag{7}$$

for the implicit midpoint method. These methods are thus well-suited for training models from data where x^n and x^{n+1} are available, as they are then computationally efficient while coming with properties that cannot be achieved with explicit integrators.

3.4. **System Identification.** The system identification approach of [4] consists of assuming a general form (1) and searching for the terms on the right-hand side, thus ideally obtaining the governing equations. That is, given data on $\dot{x}$ and x, we assume that g can be expressed by terms from a library of nonlinear functions of x, and search for a minimal linear combination $\hat{g}_\theta(x)$ of these that minimizes $\dot{x} - \hat{g}_\theta(x)$. In the absence of data on $\dot{x}$, [4] suggest to approximate this from x. In [16, 34], a modified variant of this approach is used to learn the terms of the Hamiltonian and obtain the governing equations by assuming the system has a canonical Hamiltonian formulation. The Hamiltonian function for most conservative dynamical systems will only include a few terms, which means that a small function space may suffice for determining the governing equations. In this paper, we generalize the approach by assuming a pseudo-Hamiltonian formulation, thus also having to learn damping coefficients and external forces. Furthermore, in contrast to [16], we do not assume separability of the Hamiltonian into potential and kinetic energy, or that the matrix S has the canonical form as in (2).

A clear advantage of the system identification approach compared to a black-box model such as neural networks is interpretability. Although a neural network may be accurate in prediction given sufficient data and a sufficiently large network [12, 26], it does not reveal the governing equations of the system. System identification models also hold an advantage over black-box models when predicting on data that is drawn from outside the domain of training data; or in other words, they can perform robust data extrapolation [4].

4. **Methodology.**

Our procedure for system identification is to define function spaces for the Hamiltonian H and possibly the external forces F, and train coefficients for each of the terms in the function space while also training coefficients of the damping matrix R. Given these search spaces, we will use system identification models to learn the governing equation of dynamical systems with pseudo-Hamiltonian structure. In our implementation, we have included polynomial and trigonometric terms in the function spaces, allowing for a variety of different dynamical systems.

4.1. **Pseudo-Hamiltonian system identification.**

We define the most general PHSI model by

$$\hat{g}_\theta(x,t) := (\hat{S}_\theta(x) - \hat{R}_\theta(x))\,\nabla \hat{H}_\theta(x) \, + \, \hat{F}_\theta(x,t), \tag{8}$$

where $\hat{S}_\theta$, $\hat{R}_\theta$, $\hat{H}_\theta$ and $\hat{F}_\theta$ are each modelling their corresponding term in (5). The main advantage of pseudo-Hamiltonian system identification is thus its ability to learn the true equations of a Hamiltonian system that is disturbed by both damping and external forces. Learning the true equations of the inner dynamics of a system can be difficult if it is affected by external forces with a complex form that we have little or no prior knowledge of. Separating out the forces by modeling them with $\hat{F}_\theta$ will make this possible, as we will show in the numerical experiments. $\hat{F}_\theta$ can be modeled either by a system identification model or a neural network, depending on the complexity of the true external forces F. We will demonstrate learning $\hat{F}_\theta$ through a system identification model in Section 5.3 and through a neural network in Section 5.4. In the most general case, the matrices $S(x)$ and $R(x)$ in (5) would be learned with only the assumption that they are antisymmetric resp. positive semi-definite. For the numerical experiments in this paper, we assume to know S exactly and R up to some learnable friction coefficients. This reduces the full PHSI model (8) to

$$\hat{g}_\theta(x,t) = (S - \operatorname{diag}(\hat{r}_\theta))\,\nabla \hat{H}_\theta(x) \, + \, \hat{F}_\theta(x,t), \tag{9}$$

where $\operatorname{diag}(\hat{r}_\theta)$ is the matrix with the trainable elements of $\hat{r}_\theta \in \mathbb{R}^d$ on the diagonal. Assuming that the structure matrix S is known gives both an advantage and a disadvantage: On one hand, knowing S allows us to impose the physical law of energy conservation through the Hamiltonian framework. On the other, our approach is restricted to learning models where S is prior knowledge, something that other system identification models such as [4] are not. However, in the systems we will consider, S is either of canonical form, as is assumed in the many recent works on Hamiltonian neural networks [23], or a non-canonical form that can be obtained from engineering knowledge.

4.2. **Pruning.** We have constructed a simple pruning algorithm for excluding terms during training: For every Pth epoch, every coefficient whose absolute value has taken on values smaller than a threshold ϵ for the last p epochs will be set to zero and disregarded for the rest of the training. The pruning thus narrows the search space, making it easier to find the relevant terms and accurate coefficients. The optimal choices of ϵ and p are highly problem dependent. In the numerical experiments of Section 5, we have used $p = 1$ and found this to work well, and it is this special case that is shown in Algorithm 1. In Section 6 we discuss the danger of pruning away terms with small coefficients if ϵ is not wisely chosen.

4.3. **Regularization.** To be able to separate the Hamiltonian system, the damping effects, and the external forces from each other, it is often necessary to implement regularization, especially on the forces. As discussed in Section 3.2, there is generally not uniqueness in the separation of the pseudo-Hamiltonian system into the internal and external parts. In practice, if S is known and additional structure is imposed on R so that the system would be uniquely defined in the absence of external forces, the model tends to learn the most natural representation, where only the dynamics that do not fit within the Hamiltonian structure with damping are attributed to external forces. We have observed this effect from repeated experiment, and attribute it to it being easier to learn a Hamiltonian H that conforms to the expected structure of a Hamiltonian system than to learn the entirety of dynamics represented by the force function f. However, depending on the underlying dynamics and the assumptions imposed on the model, it is often expedient or even necessary to implement regularization, especially on F. This is particularly relevant when the external forces are state-dependent and modeled by a neural network, which does not assume any specific form of the term.

We have used L_1-regularization on the coefficients of the Hamiltonian to promote sparsity of the search space, following the SINDy approach [4]. However, the regularization can negatively affect the accuracy of the trained non-zero parameters. Therefore, we choose to drop the regularization during the second half of training. The idea is that sparsity will be promoted during the first half of training, and then the remaining coefficients will be better tuned during the second half. A study of how regularization affects the structure of the learned PHSI model is provided in Appendix D.

4.4. **Loss function.** Several earlier works on system identification and HNN have assumed that the true derivatives of the system are known or can be approximated by e.g. finite differences [4, 23, 15]. Other works rely on an explicit integrator to propagate the learned system and evaluate the loss by comparing to available data on state variables [8, 14, 16]. In this paper, we train on an integration scheme (6), which to our knowledge has not previously been proposed for system identification. The main advantage is that we can use data from two successive points, thus partly averaging out the noise at no extra computational cost. We propose to use symmetric non-partitioned integrators, which depend equally on x^n and x^{n+1} when evaluating $\Psi_{\Delta t}$ in (15). E.g., if the data points x^n, $n = 1, \ldots, N$, have independent Gaussian noise with standard deviation σ, the standard deviation of the noise of $(x^n + x^{n+1})/2$ in (7) is $\sigma/\sqrt{2}$. The midpoint method is however only a second-order integrator. For all experiments, we train the PHSI models using the fourth-order

Algorithm 1 Training algorithm

```
Input variables: input data $X = [x_1, ..., x_m]$, number of epochs $E$, integrator $\Psi$,
learning rate $\eta$, batch size $b$, pruning interval $P$, pruning threshold $\varepsilon$
for $e$ in $(1, ..., E)$ do
  Create a new permutation of $p = \{p_1, ..., p_m\}$of$\{1, ..., m\}$
  for $i$ in $(1, ..., m/b)$ do
    Select $b$ samples from the shuffled dataset $X_{\text{batch}} = \{[xp_j]\}_{j=ib}^{(i+1)b}$
    Compute gradients of the loss with respect to the model parameters through
    backpropagation:
    • $G_{\Xi_H} = \frac{dL(\hat{g}_{\text{PHSI}};X_{\text{batch}},\Psi)}{d\Xi_H}$
    • $G_{\Xi_F} = \frac{dL(\hat{g}_{\text{PHSI}};X_{\text{batch}},\Psi)}{d\Xi_F}$
    • $G_{\hat{R}} = \frac{dL(\hat{g}_{\text{PHSI}};X_{\text{batch}},\Psi)}{d\hat{R}}$
    Update model parameters: $\Xi_H, \Xi_F, \hat{R} \leftarrow (\Xi_H, \Xi_F, \hat{R}) - \eta(G_{\Xi_H}, G_{\Xi_F}, G_{\hat{R}})$
  end for
  if $e//P = 0$ then
    for $\xi$ in $\{\Xi_H, \Xi_F, \hat{R}\}$ do
      if $|\xi| < \varepsilon$ then
        $\xi = 0$ for the rest of the training
      end if
    end for
  end if
end for
```

symmetric integration scheme introduced in [20]. Appendix A includes the definition of this integrator and a discussion of properties that can influence the choice of the integrator, like symmetry, symplecticity, and order.

When the numerical integrator Ψ has been chosen, the loss function is evaluated on the corresponding integration scheme:

$$
\begin{aligned}
L = &\|\frac{x^{n+1} - x^n}{\Delta t} - \Psi_{\Delta t}(\hat{g}_\theta, x^n, x^{n+1}, t^n)\|_2^2 \\
&+ \lambda_H\|\theta_H\|_1 + \lambda_F\|\theta_F\|_1 + \lambda_R\|\theta_R\|_1,
\end{aligned}
\tag{10}
$$

given for one data point x^n. Here λ_H, λ_F, and λ_R are regularization parameters, and θ_H, θ_F and θ_R denote the vectors of trainable parameters in $\hat{H}_\theta$, $\hat{F}_\theta$ and $\hat{R}_\theta$. If $\hat{F}$ is modelled by a neural network and not a system identification model, the term $\lambda_F\|\theta_F\|_1$ is replaced by $\lambda_F\|\hat{F}_\theta(\frac{x^n+x^{n+1}}{2})\|_1$. As noted above, λ_H, λ_F, and λ_R may be set to zero midway during training. Pseudocode for how the loss is computed and how the model is trained is shown in Algorithm 1.

4.5. **Initializations and hyperparameters.** When initializing the model, all coefficient parameters are set to 0.2. We want to avoid initial values of the parameters near zero, as this increases the risk of terms being removed by the pruning algorithm too early. We use the Adam optimizer and have to specify the learning rate, batch size and number of epochs to run. The other hyperparameters that have to be specified are parameters for regularization on the external forces and the damping, and the pruning parameters P and p. These are given for the specific experiments in Section 5.

4.6. **Baseline system identification model.** To evaluate the effect of imposing a pseudo-Hamiltonian structure on our model, we have tested the PHSI model against a baseline system identification (BSI) model trained with a similar strategy, including the same integrator, but without assuming a pseudo-Hamiltonian structure. This model learns a sparse representation of g in (1) from a function space similar to that used for learning H in the PHSI model, albeit with a constant term and one degree lower polynomials to reduce the search space. If we for instance want to learn a system with a Hamiltonian of polynomial terms of degree n, the search space for system identification of the Hamiltonian has $\binom{d+n}{n} - 1$ terms. A system identification model directly learning a representation of g will have a search space of $d \cdot \binom{d+n-1}{n-1}$, which is more than $\frac{dn}{d+n}$ times as many terms. Since this term can grow large for large values of d and n, the search space of g can be much larger than that of the Hamiltonian H, making it more difficult to learn through system identification. This difference in structure can give the BSI model a disadvantage compared to the PHSI model.

5. **Experiments.** To assess the performance of the PHSI model, we compare it against three other models: the SINDy model of [4], learning g in (1); the sparse symplectically integrated neural networks (SSINN) of [16], which learns a separable Hamiltonian, assuming the structure in (2); and the baseline model (BSI) presented in Section 4.6, learning g in (1). For each test, we train the relevant models on two data sets: one noise-free and one with Gaussian noise: $x^i_{\text{noise}} = x^i + \epsilon^i$ for all x^i, where $\epsilon^i \sim N(0, \sigma^2)$. When choosing the polynomial degree of the search space, SINDy and BSI will use one polynomial degree lower than PHSI, since they train on the derivative terms of the Hamiltonian. The data sets consist of trajectories with random initialization.

The Adam optimizer is used with a weight decay constant of 10^{-4}. The learning rate is chosen from the following search space: $\{10^{-2}, 3 \cdot 10^{-3}, 10^{-3}\}$. The training data is shuffled for all experiments, and the integrator is the fourth-order symmetric integration scheme introduced in [20]; see Appendix A for details on this. The batch size is 32. For the PHSI and BSI models, polynomial coefficients have an initial value of 0.2, and trigonometric coefficients (amplitude and frequency) have an initial value of 1.

5.1. **Learning a separable Hamiltonian system.** We first test our methodology on data obtained from a pure Hamiltonian system. Thus we can benchmark against existing methods while demonstrating how our proposed framework facilitates assumptions being imposed. We consider the Hénon–Heiles system for describing the two-dimensional chaotic motion of stars around a galactic center [25]. It is a canonical system of the form (2) with $n = 2$, and the Hamiltonian is defined as

$$H(q, p) = \frac{1}{2}(q_1^2 + q_2^2 + p_1^2 + p_2^2) + q_1^2 q_2 - \frac{1}{3} q_2^3.$$

Because this is a separable Hamiltonian system, it can be modeled by SSINN.

To ensure a fair comparison, we train on the data generated and used in [16], although with more noise. The data consists of 3000 pairs of q, p at $t = 0$ and $t = 0.1$. The noisy data have $\sigma = 0.02$, which corresponds to approximately 3% of the maximum absolute values in the data. For the results reported here, the learning rate is $3 \cdot 10^{-3}$, and regularization is not used. The PHSI and BSI models are each trained for 60 epochs. The P-value and ϵ-value in the pruning algorithm

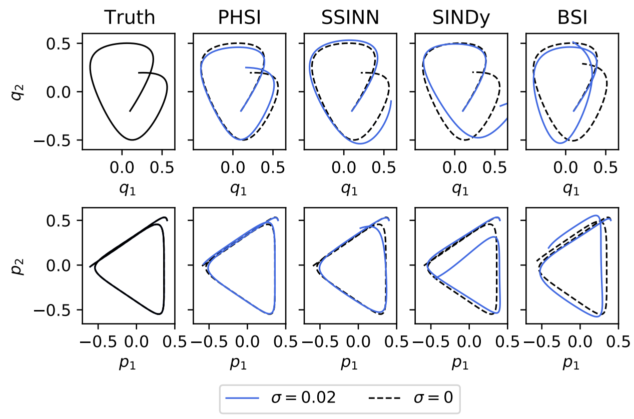

FIGURE 1. Comparison of simulated trajectories for the Hénon–Heiles system, with initial value $(q, p) = (0.1, -0.2, 0.4, 0.5)$, from $t = 0$ to $t = 10$.

(1) are set to 5 and 0.05, respectively. The search space for PHSI and SSINN is third-degree polynomials, while for SINDy and BSI it is second-degree polynomials.

Figure 1 shows example trajectories obtained from integrating the different learned systems in time. When training on noise-free data, all models learn the true coefficients up to 10^{-2}, while noise affects the different methods to different degrees. On the noisy data, PHSI outperforms SSINN by one order of magnitude when comparing the L_2-error of the trajectories, as depicted in Figure 2. PHSI and BSI both perform better than SSINN and SINDy, suggesting that the fourth-order symmetric integrator handles the noise well. The superior performance of PHSI over BSI suggests that imposing the Hamiltonian structure gives better performance yet. The SINDy and BSI models could not be converted to Hamiltonian functions when trained on noisy data, and the solution of the SINDy model tends to become unstable over long time periods. Table 1 compares the coefficients learned from training on the noisy data set for the two models imposing a Hamiltonian structure.

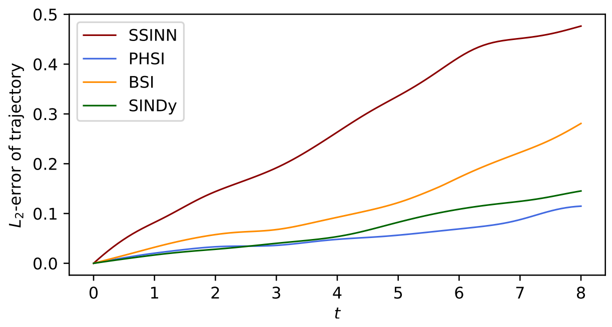

FIGURE 2. The average L_2-error of the trajectories obtained with 10 different random initial conditions, from the different models of the Hénon–Heiles system trained on noisy data with $\sigma = 0.02$, compared to trajectories simulated from the exact system,

TABLE 1. Learned coefficients in the Hamiltonian for the Hénon–Heiles system.

	q_1^2	q_2^2	$q_1^2 q_2$	q_2^3	p_1^2	p_2^2
True Value	0.5	0.5	1	-0.333	0.5	0.5
PHSI	0.509	0.495	1.009	-0.338	0.501	0.477
SSINN	0.421	0.378	0.569	-0.195	0.484	0.443
	$q_2 p_1^2$	q_2	$q_1 q_2$	p_2	$p_1 p_2$	
True Value	0	0	0	0	0	
PHSI	0.124	0	0	0	0	
SSINN	N/A	0.004	0.002	-0.006	-0.006	

5.2. **Learning a non-separable Hamiltonian system.** A strength of PHSI is that it can learn non-separable Hamiltonians as well. We demonstrate this by testing its ability to learn a finite-dimensional nonlinear Schrödinger system, as considered in [50]. This is a canonical system (3) with $d = 2$ and the Hamiltonian

$$\begin{aligned} H(q,p) = &\frac{1}{4}(q_1^2 + p_1^2)^2 + \frac{1}{4}(q_2^2 + p_2^2)^2 \\ &- q_1^2 q_2^2 - p_1^2 p_2^2 + q_1^2 p_2^2 + q_2^2 p_1^2 - 4 q_1 q_2 p_1 p_2. \end{aligned}$$

We use two sets of training data: one clean and one noisy with $\sigma = 5 \cdot 10^{-4}$. The training sets consist of 30 trajectories consisting of 100 points with step size 0.01, each randomly initialized from a uniform distribution between 0 and 1. We have trained the PHSI, SINDy and BSI models, and compare the performances of these. For the PHSI model, the learning rate is 0.01, and regularization is not used. The model is trained for 100 epochs. We use $P = 20$ and $\epsilon = 0.05$ in the pruning algorithm. The search space for PHSI is fourth-degree polynomials, while for SINDy and BSI it is third-degree polynomials.

PHSI and SINDy are able to learn the true equations of the system on the noise-free data up to a precision of 10^{-5}, while BSI is able to learn to a precision of 10^{-3}. On the noisy data, PHSI learns the true equations up to a precision of 10^{-1} and excluding 53 out of 58 terms correctly by the pruning algorithm. SINDy only learns the terms up to a precision of 1, and BSI does not even achieve this precision and also exclude many terms incorrectly during pruning. Figure 4 confirms a more accurate representation of the true system equations in the PHSI model than SINDy when trained on noisy data. Neither the coefficients learned by SINDy nor BSI could be converted to a separable Hamiltonian.

5.3. **Learning a pseudo-Hamiltonian system.** Consider a mass-spring system with damping and external forces. The Hamiltonian is given by

$$H(q,p) = \frac{k}{2} q^2 + \frac{1}{2m} p^2,$$

where k is the stiffness constant of the spring and m is the mass. Furthermore, we have damping and an external force affecting the momentum, so that a pseudo-Hamiltonian formulation (5) of the system is given by

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -c \end{bmatrix} \begin{bmatrix} kq \\ \frac{1}{m} p \end{bmatrix} + \begin{bmatrix} 0 \\ \alpha \sin(\omega t) \end{bmatrix}. \tag{11}$$

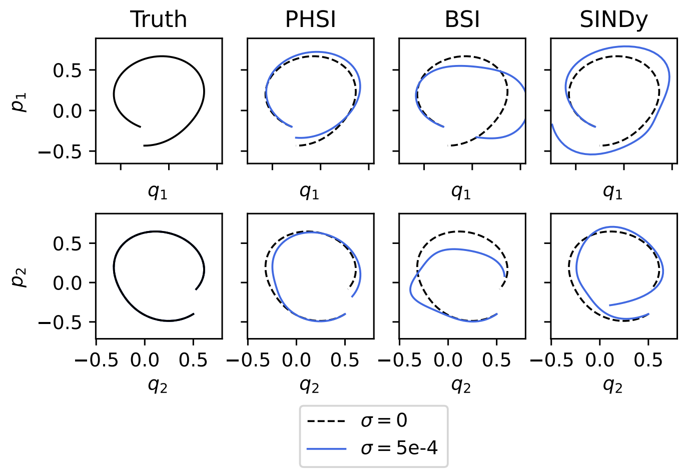

FIGURE 3. Phase portraits showing the trained models' trajectories next to the ground truth trajectory, for the nonlinear Schrödinger system. All models are trained on the two training sets, one clean data set and one noisy with $\sigma = 10^{-4}$. The initial values are $(q, p) = (-0.3, 0.5, -0.2, -0.4)$.

We let $k = m = 1$, $\alpha = 2$, $\omega = 0.5$ and $c = 0.3$. The data consists of 50 trajectories from time 0 to 10, with time step 0.1, and the noisy data set has $\sigma = 0.2$.

We train three system identification models, PHSI, BSI, and SINDy, and also a PHNN model as described in [15]. For the PHSI model, we train a third-degree polynomial function space for the Hamiltonian, while the function space for the external force is a third-degree polynomial as well as trigonometric functions where both amplitude and frequency are trainable parameters, and the force is assumed to be strictly time-dependent and only directly affecting the momentum. In addition, we assume that damping only directly affects the momentum, so that learning the matrix R in this case reduces to learning the friction coefficient c. Both the BSI and SINDy models are trained on a third-degree polynomial (including a constant) plus trigonometric functions. The SINDy model has a disadvantage as it can only learn autonomous systems. To still be able to use it, we convert the system into an autonomous one by adding time as a variable to (11): $\dot{t} = 1$. The PHSI, BSI, and PHNN models are run for 150 epochs, with a learning rate $5 \cdot 10^{-3}$. We set $P = 20$ and $\epsilon = 0.05$ in the pruning algorithm. L_1-regularization is used for the PHSI model on the forces and the Hamiltonian function: $\lambda_H = 0.1$ and $\lambda_F = 0.01$ in (10). The external force of the PHNN model has a regularization parameter of 0.1.

Predictions resulting from the models are shown in Figure 5 and coefficients are given in Table 2. From Figure 6 we see that PHSI clearly outperforms PHNN on predictions beyond the time period of the training data, which is expected since the external force is time-dependent. Even on noisy data, the PHSI model is able to separate the inner dynamics, the damping effects and the external forces from each other, while the BSI model did not learn terms that can be separated into inner

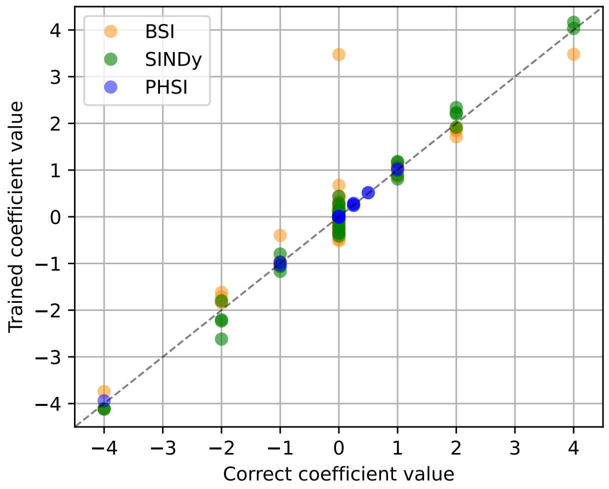

FIGURE 4. The trained coefficients of the PHSI, BSI, and SINDy models of the nonlinear Schrödinger system, plotted against the respective true values. A perfectly trained model will only have points along the dotted line. The models are trained on the noisy data set. Note that PHSI learns the Hamiltonian function while SINDy learns the right-hand side g of (1) and hence they will not learn the same coefficients for corresponding terms.

and outer dynamics in any obvious way. This provides a qualitative argument for using a model that assumes a pseudo-Hamiltonian formulation.

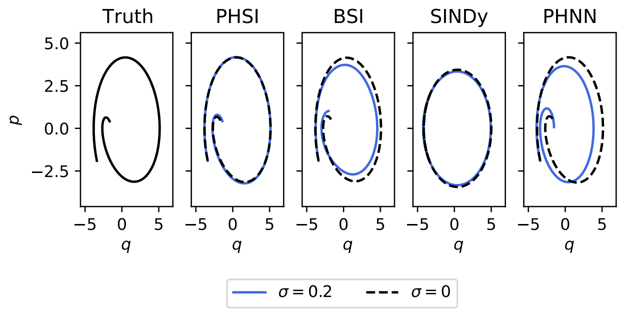

FIGURE 5. Comparison between the phase portraits obtained from integrating the exact forced and damped mass-spring system and the learned models from time 0 to 10. The initial value is $(q, p) = (-3.4, -1.9)$.

5.4. **Hybrid model combining system identification with a neural network.** In reality, many systems will be affected by external forces for which it can be difficult to find analytic terms. The final system to be studied consists of N tanks connected by M pipes, also considered in [20], with leaks that can be viewed

TABLE 2. Learned coefficients for the forced and damped mass-spring problem. q and p are multiplied with the trained coefficients while c, α, ω and $const.$ are themselves the trainable parameters. Empty entries mean that the model does not learn that term.

Trained PHSI parameters					
	q^2	p^2	c	α	ω
True value	0.5	0.5	0.3	2	0.5
$\hat{H}$	0.473	0.484			
$\hat{R}$			0.300		
$\hat{F}$	0	0	0	1.984	0.505
Trained BSI and SINDy parameters					
	q	p	$const.$	α	ω
True $\hat{\dot{q}}$	0	1	0	0	0
BSI $\hat{\dot{q}}$	0	0.991	0	0	0
SINDy $\hat{\dot{q}}$	0	0.983	0	0	0
True $\hat{\dot{p}}$	-1	-0.3	0	2	0.5
BSI $\hat{\dot{p}}$	-0.980	-0.278	0	1.999	0.506
SINDy $\hat{\dot{p}}$	-0.548	0	0.241		

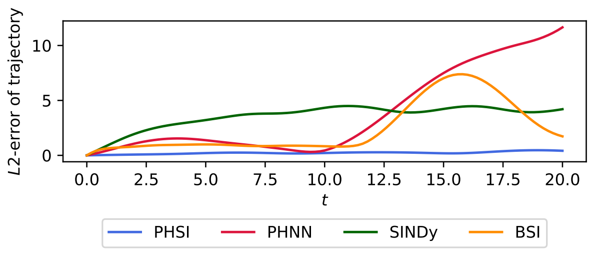

FIGURE 6. Average L_2-error of 30 simulated trajectories of the forced and damped mass-spring system, with random initial conditions.

as external forces. The incidence matrix $B \in \mathbb{R}^{M\times N}$ describes how the pipes and tanks are connected. The friction in the pipes is assumed to depend linearly on the flow, and the total energy is given by

$$H(\phi, \mu) = \sum_{i}^{M} \frac{1}{2J_i}\phi_i^2 + \sum_{j}^{N} \frac{g\rho}{2A_j}\mu_j^2, \tag{12}$$

where ϕ_i is the flow in pipe i scaled by a factor J_i depending on the density of the fluid and the dimension of the pipe, μ_j is the volume of the fluid in tank j, g is the gravitational constant, ρ is the density of the fluid, and A_j is the footprint of tank

j. We thus have the pseudo-Hamiltonian formulation

$$\begin{bmatrix}\dot{\phi}\\ \dot{\mu}\end{bmatrix} = \begin{bmatrix}-\text{diag}(r_p) & B^T\\ B & 0_{N\times N}\end{bmatrix}\begin{bmatrix}\frac{\partial H}{\partial \phi}\\ \frac{\partial H}{\partial \mu}\end{bmatrix} + \begin{bmatrix}0_M\\ f(\phi,\mu)\end{bmatrix}, \tag{13}$$

where $r_p \in \mathbb{R}^M$ contains the friction coefficients relating to each pipe. When simulating the system to generate the training data, we set $\rho = 1$, $J_i = 0.02\ \forall i$, $A_j = 1\ \forall j$, $r_p = (0.03, 0.03, 0.09, 0.05, 0.05)$. A leak in the fourth tank is described by $f(\phi,\mu) = (0,0,0,-10\min(0.3,\max(\mu_4,0.3)))$. Initial conditions are uniformly drawn, $-1 < x^i < 1$, for all i.

We compare SINDy and BSI against a hybrid PHSI model that only aims to find analytic terms for the internal dynamics. It assumes that the external force only affects the fourth tank, and models this by a neural network. This consists of three hidden layers, 100 nodes per layer, and the ReLU activation function in each layer. The training data consists of 60 trajectories from $t = 0$ to $t = 0.5$, with time step 0.01. The PHSI and BSI models are trained for 100 epochs, with learning rate $3 \cdot 10^{-2}$ and $P = 10$ and $\epsilon = 0.05$. For the PHSI model, L_1 regularization is used on the external forces and the Hamiltonian with $\lambda_H = 0.5, \lambda_F = 0.001$ in (10). The search space for $\hat{H}_\theta$ is second-degree polynomials. The BSI and SINDy models has a search space of first-degree polynomials.

Figure 7 provides example trajectories obtained from the models and Figure 8 shows the error averaged over predictions from 30 random initial conditions. The PHSI model learned the true Hamiltonian up to 10^{-1} precision, as well as learning a precise neural network model of the leak. The learned PHSI parameters are shown in Table 3. BSI and SINDy did not learn terms that could be converted to a Hamiltonian function and have no clear separation of the internal system from the external forces. This indicates that they are not able to accurately learn the true system, and instead learn polynomial approximations that approximate the system well only within the domain of the training data. The lower plots in Figure 7 confirm this. When testing the learned models on initial conditions well outside the range of the training data, PHSI greatly outperforms the other models.

Table 3. Learned coefficients for the tank system on the noisy data. $x_1^2, \dots, x_9^2$ are multiplied with the trained coefficients while $r_1, \dots, r_5$ are themselves the trainable parameters.

	x_1^2	x_2^2	x_3^2	x_4^2	x_5^2	x_6^2	x_7^2
True value	25	25	25	25	25	4.905	4.905
PHSI	24.94	24.98	24.98	24.95	24.97	4.930	4.960
	x_8^2	x_9^2	r_1	r_2	r_3	r_4	r_5
True value	4.905	4.905	0.03	0.03	0.09	0.05	0.05
PHSI	4.890	4.930	0.031	0.029	0.086	0.051	0.041

6. Pruning and small coefficients.

The pruning technique outlined in Section 4.2 and Algorithm 1 introduces the hyperparameters P, ϵ and p. These parameters can greatly affect the training, and they should therefore be set with care and depending on the problem. In particular, no coefficent smaller than ϵ can be learned,

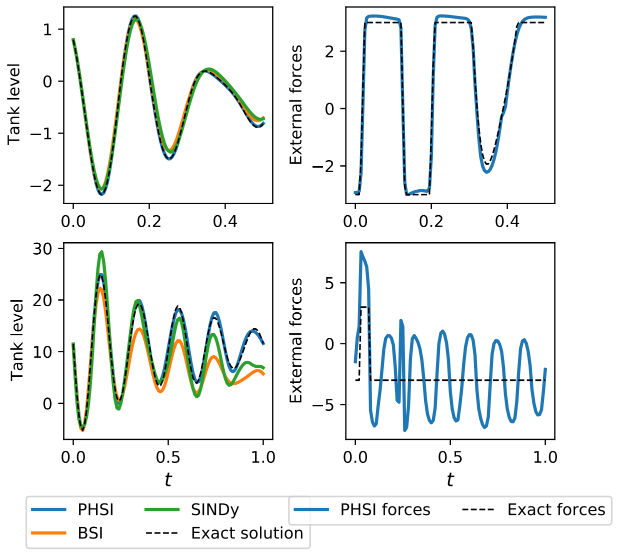

FIGURE 7. Simulations of the tank system. *Left:* The volume of the fluid in the leaky fourth tank simulated from the exact system and the different models. *Right:* The leak approximated by the neural network in the hybrid PHSI model, compared to the exact solution. The upper plots have initial conditions within the distribution of the training data: $(\phi, \mu) = (-0.4, 0, 0.5, 0, 0.2, 0 - 0.6, -0.5, 0.5)$. The lower plots show extrapolation in time and space; time from 0 to 1 and initial state values $(\phi, \mu) = (10, 19, 4, 19, 7, 9, 17, 9, 11)$.

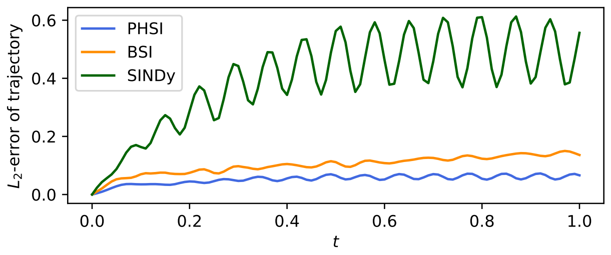

FIGURE 8. Average L_2-error of 30 sets of simulated future tank volumes and pipe flows in the tank system, trained on noisy data with $\sigma = 0.005$.

so it should not be set too high. On the other hand, the pruning becomes more challenging with lower values of ϵ.

Thus it is clear that the PHSI method, like other system identification methods, will struggle to learn terms with small coefficients. To test this, we apply the same methods as in Section 5.3 on a slightly skewed mass-spring system with the same damping and external force. That is, we modify the Hamiltonian to be

$$H(q,p) = \frac{1}{2}q^2 + \frac{1}{2}p^2 + \gamma qp, \tag{14}$$

for $\gamma = \{0.1, 0.07, 0.05, 0.03, 0.01, 0.007, 0.005, 0.003, 0.001\}$ and train and test the methods on the resulting system. We try $\epsilon = \{0.05, 0.01, 0.001\}$, and use otherwise the same hyperparameters for the pruning as in Section 5.3, where $\epsilon = 0.05$ and $\gamma = 0$. When $\epsilon = 0.05$ or $\epsilon = 0.01$, the PHSI method prunes away all terms that should be zero, whether it trains on exact or noisy data, for all choices of γ. However, when $\epsilon = 0.001$, it fails to prune away some of the terms that should be zero when training on the noisy data. E.g., when $\gamma = 0.003$ we get

$$\begin{aligned}
&\hat{H}_\theta(q,p) = 0.4987q^2 + 0.5004p^2 + 0.0028qp,\\
&\hat{c} = 0.2978,\\
&\hat{f}_\theta(t) = 1.9767\sin(0.4993t)
\end{aligned}$$

on the noise-free data, but

$$\begin{aligned}
&\hat{H}_\theta(q,p) = 0.0044p^3 + 0.4845q^2 + 0.5075p^2 - 0.0196qp + 0.0012q - 0.0098p,\\
&\hat{c} = 0.2709,\\
&\hat{f}_\theta(t) = 1.9334\sin(0.4951t)
\end{aligned}$$

on the noisy data.

As we see in Table 4, the PHSI model learns a non-zero approximation of γ when it trains on noise-free data and $\gamma > \epsilon$, with the sole exception $\gamma = 0.005, \epsilon = 0.001$. When the data is noisy, we see in Table 5 that the method prunes away the qp term when $\gamma = 0.07, \epsilon = 0.05$ and $\gamma = 0.005, \epsilon = 0.001$, but otherwise learns non-zero approximations of $\gamma > \epsilon$.

Table 4. The learned approximations of γ in (14), for different γ and different pruning tresholds ϵ, trained on noise-free data.

$\epsilon \backslash \gamma$	0.1	0.07	0.05	0.03	0.01	0.007	0.005	0.003	0.001
0.05	0.1012	0.0718	0	0	0	0	0	0	0
0.01	0.1000	0.0717	0.0501	0.0310	0	0	0	0	0
0.001	0.1006	0.0711	0.0489	0.0301	0.0102	0.0051	0	0.0028	0

Table 5. The learned approximations of γ in (14), for different γ and different pruning tresholds ϵ, trained on data with added Gaussian noise with standard deviation $\sigma = 0.2$.

$\epsilon \backslash \gamma$	0.1	0.07	0.05	0.03	0.01	0.007	0.005	0.003	0.001
0.05	0.0974	0	0	0	0	0	0	0	0
0.01	0.1001	0.0485	0.0515	0.0412	0	0	0	0	0
0.001	0.0833	0.0754	0.0372	0.0399	0.0034	0.0093	0	-0.0196	-0.0062

7. Conclusion.

This paper presents several advances in the identification of physical systems that can be modeled as ODEs. For one, we demonstrate improved learning of Hamiltonian systems, compared to standard methods and a recent model specifically developed for such systems. Secondly, we develop the methodology further to be applicable to the much wider class of pseudo-Hamiltonian systems. Lastly, we show how we can incorporate neural network models in our method to separate out external forces that are difficult to identify by analytic terms. In all experiments, the PHSI model performs especially well on noisy data. We attribute this both to the model assumption of an internal Hamiltonian structure and a training strategy that automatically averages out some of the noise.

We have not emphasized computational cost in our numerical examples, and have not yet made a significant effort to optimize our implementation of PHSI or BSI. Compared to SINDy, our models use significantly longer time to train to convergence. This will be addressed in future research, as will extensions to more complex systems, including partial differential equations.

Acknowledgments. The authors thank Vegard Antun and Alexander Stasik for fruitful discussions and insightful comments.

REFERENCES

[1] F. Alet, D. Doblar, A. Zhou, J. Tenenbaum, K. Kawaguchi, and C. Finn. Noether networks: meta-learning useful conserved quantities. *Advances in Neural Information Processing Systems*, 34:16384–16397, 2021.

[2] V. I. Arnold. *Mathematical methods of classical mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1989. Translated from the Russian by K. Vogtmann and A. Weinstein.

[3] P. Benner, P. Goyal, and P. Van Dooren. Identification of port-Hamiltonian systems from frequency response data. *Systems Control Lett.*, 143:104741, 10, 2020.

[4] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[5] J. R. Cash and A. Singhal. High order methods for the numerical solution of two-point boundary value problems. *BIT*, 22(2):184–199, 1982.

[6] J. R. Cash and A. Singhal. Mono-implicit Runge-Kutta formulae for the numerical integration of stiff differential systems. *IMA J. Numer. Anal.*, 2(2):211–227, 1982.

[7] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, pages 6572–6583, 2018.

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

[9] K. Cherifi, P. Goyal, and P. Benner. A non-intrusive method to inferring linear port-Hamiltonian realizations using time-domain data. *Electron. Trans. Numer. Anal.*, 56:102–116, 2022.

[10] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[11] M. Cranmer, A. Sanchez Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33:17429–17442, 2020.

[12] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[13] M. David and F. Méhats. Symplectic learning for Hamiltonian neural networks. *arXiv preprint arXiv:2106.11753*, 2021.

[14] S. A. Desai, M. Mattheakis, and S. J. Roberts. Variational integrator graph networks for learning energy-conserving dynamical systems. *Physical Review E*, 104(3):035310, 2021.

[15] S. A. Desai, M. Mattheakis, D. Sondak, P. Protopapas, and S. J. Roberts. Port-Hamiltonian neural networks for learning explicit time-dependent dynamical systems. *Phys. Rev. E*, 104:034312, 9 2021.

[16] D. DiPietro, S. Xiong, and B. Zhu. Sparse symplectically integrated neural networks. *Advances in Neural Information Processing Systems*, 33:6074–6085, 2020.

[17] T. Duong and N. Atanasov. Hamiltonian-based neural ODE networks on the SE(3) manifold for dynamics learning and control. In *Robotics: Science and Systems (RSS)*, 2021.

[18] T. Duong and N. Atanasov. Adaptive control of SE(3) Hamiltonian dynamics with learned disturbance features. *IEEE Control Syst. Lett.*, 6:2773–2778, 2022.

[19] S. Eidnes and K. O. Lye. Pseudo-Hamiltonian neural networks for learning partial differential equations. *arXiv preprint, arXiv:2304.14374*, 2023.

[20] S. Eidnes, A. J. Stasik, C. Sterud, E. Bøhn, and S. Riemer-Sørensen. Pseudo-Hamiltonian neural networks with state-dependent external forces. *Phys. D*, 446:Paper No. 133673, 13, 2023.

[21] U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton. Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A*, 478(2260):20210904, 2022.

[22] M. Finzi, K. A. Wang, and A. G. Wilson. Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints. *Advances in neural information processing systems*, 33:13880–13889, 2020.

[23] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[24] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2006. Structure-preserving algorithms for ordinary differential equations.

[25] M. Hénon and C. Heiles. The applicability of the third integral of motion: Some numerical experiments. *Astronom. J.*, 69:73–79, 1964.

[26] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[27] P. Jin, Z. Zhang, I. G. Kevrekidis, and G. E. Karniadakis. Learning Poisson systems and trajectories of autonomous systems via Poisson neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

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

[29] K. Kaheman, J. N. Kutz, and S. L. Brunton. Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A*, 476(2242):20200279, 2020.

[30] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

[31] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[32] S. Klus, F. Nüske, S. Peitz, J.-H. Niemann, C. Clementi, and C. Schütte. Data-driven approximation of the Koopman generator: Model reduction, system identification, and control. *Phys. D*, 406:132416, 15, 2020.

[33] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

[34] K. Lee, N. Trask, and P. Stinis. Structure-preserving sparse identification of nonlinear dynamics for data-driven modeling. *Proceedings of Mathematical and Scientific Machine Learning*, 190:65–80, 2022.

[35] Z. Liu and M. Tegmark. Machine learning conservation laws from trajectories. *Phys. Rev. Lett.*, 126(18):Paper No. 180604, 6, 2021.

[36] L. Ljung. *System identification: theory for the user*. Prentice Hall Information and System Sciences Series. Prentice Hall, Inc., Englewood Cliffs, NJ, 1987.

[37] Z. Long, Y. Lu, and B. Dong. PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.*, 399:108925, 17, 2019.

[38] J. E. Marsden and T. S. Ratiu. *Introduction to mechanics and symmetry*, volume 17 of *Texts in Applied Mathematics*. Springer-Verlag, New York, second edition, 1999. A basic exposition of classical mechanical systems.

[39] T. Matsubara, A. Ishikawa, and T. Yaguchi. Deep energy-based modeling of discrete-time physics. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13100–13111. Curran Associates, Inc., 2020.

[40] T. Matsubara and T. Yaguchi. Finde: Neural differential equations for finding and preserving invariant quantities. *arXiv preprint arXiv:2210.00272*, 2022.

[41] R. I. McLachlan, G. R. W. Quispel, and N. Robidoux. Geometric integration using discrete gradients. *R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci.*, 357(1754):1021–1045, 1999.

[42] R. Morandin, J. Nicodemus, and B. Unger. Port-Hamiltonian dynamic mode decomposition. *arXiv preprint arXiv:2204.13474*, 2022.

[43] H. Noren. Learning Hamiltonian systems with mono-implicit Runge–Kutta methods. *arXiv preprint, arXiv:2303.03769*, 2023.

[44] C. Offen and S. Ober-Blöbaum. Symplectic integration of learned Hamiltonian systems. *Chaos*, 32(1):Paper No. 013122, 13, 2022.

[45] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[46] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[47] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

[48] H. Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, 2017.

[49] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[50] M. Tao. Explicit symplectic approximation of nonseparable hamiltonians: Algorithm and long time performance. *Physical Review E*, 94(4):043303, 2016.

[51] S.-M. Udrescu and M. Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[52] W. M. G. van Bokhoven. Efficient higher order implicit one-step methods for integration of stiff differential equations. *BIT*, 20(1):34–43, 1980.

[53] A. Van Der Schaft. Port-Hamiltonian systems: an introductory survey. In *Proceedings of the international congress of mathematicians*, volume 3, pages 1339–1365. Citeseer, 2006.

[54] A. Van Der Schaft and D. Jeltsema. Port-Hamiltonian systems theory: An introductory overview. *Foundations and Trends in Systems and Control*, 1(2-3):173–378, 2014.

[55] A. van der Schaft and B. Maschke. The Hamiltonian formulation of energy conserving physical systems with external ports. *AEU. Archiv für Elektronik und Übertragungstechnik*, 49(5-6):362–371, 1995.

[56] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Comput. Surv.*, 55(4), nov 2022.

[57] H. Yoshida. Construction of higher order symplectic integrators. *Phys. Lett. A*, 150(5-7):262–268, 1990.

[58] Z. Zhang, Y. Shin, and G. E. Karniadakis. GFINNs: GENERIC formalism informed neural networks for deterministic and stochastic dynamical systems. *Philos. Trans. Roy. Soc. A*, 380(2229):Paper No. 20210207, 21, 2022.

[59] Y. D. Zhong, B. Dey, and A. Chakraborty. Dissipative SymODEN: Encoding Hamiltonian dynamics with dissipation and control into deep learning. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[60] Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ODE-net: Learning Hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020.

[61] A. Zhu, P. Jin, and Y. Tang. Deep Hamiltonian networks based on symplectic integrators. *arXiv preprint, arXiv:2004.13830*, 2020.

[62] A. Zhu, P. Jin, B. Zhu, and Y. Tang. On numerical integration in neural ordinary differential equations. In *International Conference on Machine Learning*, pages 27527–27547. PMLR, 2022.

Received xxxx 20xx; revised xxxx 20xx; early access xxxx 20xx.

Appendix A. **On numerical integration schemes for learning dynamical system models.** Consider a known ODE (1). A numerical integration scheme

$$\frac{x^{n+1}-x^n}{\Delta t} = \Psi_{\Delta t}(g, x^n, x^{n+1}, t^n) \tag{15}$$

can be used to find the approximate solution $x^{n+1} \approx x(t+\Delta t)$ given $x^n \approx x(t)$. Such a scheme is *implicit* if $\Psi_{\Delta t}$ depends on x^{n+1} or intermediate steps and *explicit* otherwise. E.g., we have the explicit (forward) Euler method by $\Psi_{\Delta t}(g, x^n, x^{n+1}, t^n) = g(x^n, t^n)$ and the implicit (backward) Euler method by $\Psi_{\Delta t}(g, x^n, x^{n+1}, t^n) = g(x^{n+1}, t^n + \Delta t)$.

The main drawback of using an implicit integrator for numerical integration is computational cost. An implicit scheme generally results in a system of equations that have to be solved using a root-finding algorithm like Newton's method. However, when we have available successive data points, other considerations should be taken, and we argue that we should use the available data to alleviate the effect of noise. Thus we suggest to use an integration scheme that relies equally on the two successive data points, averaging out the noise from two measurements. To that end, we consider a training strategy similar to that of [39, 28, 13, 20, 43], where we use a loss function evaluated on the integration scheme:

$$\mathcal{L} = \left\| \frac{x^{n+1}-x^n}{\Delta t} - \Psi_{\Delta t}(\hat{g}_\theta, x^n, x^{n+1}, t^n) \right\|_2, \tag{16}$$

given for one data point x^n and barring regularization.

By contrast, works like [7, 8, 16, 14] use a loss function evaluating the difference between a data point and what is obtained doing integration:

$$\mathcal{L} = \|x^{n+1} - \hat{x}^{n+1}\|_2 = \|x^{n+1} - (\hat{x}^n + \Delta t \Psi_{\Delta t}(\hat{g}_\theta, \hat{x}^n, \cdot, t^n))\|_2. \tag{17}$$

Note that the approaches are equivalent if the same explicit integrator is used and only one integration step is done between the data points so that $\hat{x}^n = x^n$ in (17).

When wishing to approximate the ODE (1) using two available successive data points $x^n = x(t^n)$ and $x^{n+1} = x(t^n + \Delta t)$, we say that a discretization method (15) is explicit if $\Psi_{\Delta t}(g, x^n, x^{n+1}, t^n)$ depends explicitly only on the two points x^n and x^{n+1} in addition to t^n and Δt, i.e. without relying on implicit intermediate steps. Note that this is different from the definition of an explicit integrator, which cannot depend on x^{n+1}. This class of explicit discretization methods includes the methods that lead to explicit integrators, but also implicit integrators that do not require numerical integration to obtain intermediate steps, like the implicit midpoint method (7). Within the larger class of Runge–Kutta methods, explicit discretization methods correspond to the class called mono-implicit Runge–Kutta methods [52, 6]. This class does not include higher-order symplectic methods like the Gauss–Legendre methods of orders four and six, which require intermediate steps to be found implicitly.

For our numerical experiments, we have used the fourth-order symmetric integrator first proposed in [20], where it is formally given for time-independent g. Also including time-dependency, the method is given by

$$\begin{aligned}
&\Psi_{\Delta t}(g, x^n, x^{n+1}, t^n) = \\
&\quad \frac{1}{2} g\left(\frac{x^n + x^{n+1}}{2} - \frac{\sqrt{3}}{6}\Delta t\, g(c_1 x^n + c_2 x^{n+1}, t^n + c_2\Delta t),\, t^n + c_1\Delta t\right) \\
&\quad + \frac{1}{2} g\left(\frac{x^n + x^{n+1}}{2} + \frac{\sqrt{3}}{6}\Delta t\, g(c_2 x^n + c_1 x^{n+1}, t^n + c_1\Delta t),\, t^n + c_2\Delta t\right),
\end{aligned} \tag{18}$$

for $c_1 = \frac{1}{2} - \frac{\sqrt{3}}{6}$ and $c_2 = \frac{1}{2} + \frac{\sqrt{3}}{6}$. An equivalent definition can be represented by the Butcher tableau

$$\begin{array}{c|cccc}
\frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & 0 & -\frac{\sqrt{3}}{6} & \frac{1}{4} \\
\frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} - \frac{\sqrt{3}}{12} & 0 & 0 & \frac{1}{4} - \frac{\sqrt{3}}{12} \\
\frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{12} & 0 & 0 & \frac{1}{4} + \frac{\sqrt{3}}{12} \\
\frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{\sqrt{3}}{6} & 0 & \frac{1}{4} \\
\hline
 & \frac{1}{2} & 0 & 0 & \frac{1}{2}
\end{array}$$

We see that it satisfies the conditions for fourth order, as given in Table III.1.1 of [24] (since all symmetric methods have even order, it is sufficient to check the third order conditions). We see also that it does not satisfy the conditions for symplecticity, as given in Theorem VI.4.3 of [24].

Several papers have suggested using symplectic integrators for learning Hamiltonian systems [8, 16, 14], but a solid theoretical argument for this has to our understanding not yet been provided. In numerical integration of a Hamiltonian system, symplectic integrators are often viewed as favorable for long-term integration, since they guarantee exact preservation of an approximated Hamiltonian, and preservation of the exact Hamiltonian within a bound [24]. This provides qualitative properties, e.g. a guarantee for stability, that do not translate directly to the inverse problem of learning a system from known data.

The integrator used in [16] is Yoshida's fourth-order method, which is also suggested for HNN in [14], although neither paper refers to it by that name or cite the proper reference [57]. This method is a partitioned Runge–Kutta method that is explicit for separable systems, but neither explicit nor mono-implicit for non-separable systems. Thus it would not be a good choice if one cannot assume the system to be separable, but [16, 14] do assume separability and argue for using this integrator because it is a fourth-order symplectic method. An advantage of using (17) with an explicit integrator is that one can do several steps between each training data point without having to rely on a root-finding algorithm like Newton's method, and [16] do indeed take advantage of this to do several shorter integration steps for each training step. This property means that given noise-free data, one can achieve arbitrarily high accuracy on the integration from x^n to $\hat{x}^{n+1}$, limited only by computational resources and time. However, it is generally less computationally efficient to use many steps than to use a higher-order integrator, and in the presence of noise, using more steps does not alleviate the issue of getting the noise from only one data point.

Appendix B. Numerical comparison of integration schemes.

We set up PHSI models with different integrators in the loss function (10) and everything else the same, to test how well different integrators with different properties affect the results. The integrators are

- the forward Euler method (Euler)
- the implicit midpoint method (7) (Midpoint)
- the classic fourth-order Runge–Kutta method (RK4)
- the symmetric fourth-order Runge–Kutta method (18) (SRK4)
- Cash and Singhal's symmetric sixth-order Runge–Kutta method [5] (SRK6)
- Yoshida's fourth order partitioned Runge–Kutta method (PRK4)

and their properties are summarized in Table 6.

Table 6. Properties of integrators. PRK4 is explicit, mono-implicit and symplectic for separable systems but not for non-separable systems.

Integrator	order	g eval's	explicit	mono-implicit	symmetric	symplectic
Euler	1	1	yes	yes	no	no
Midpoint	2	1	no	yes	yes	yes
RK4	4	4	yes	yes	no	no
SRK4	4	4	no	yes	yes	no
SRK6	6	5	no	yes	yes	no
PRK4	4	7	yes/no	yes/no	yes	yes/no

B.1. **Hamiltonian system: Hénon–Heiles.** We train the models on trajectories from $t = 0$ to $t = 10$ with

- sampling time 1 and 250 samples, i.e. 25 trajectories, without noise;
- sampling time 1 and 250 samples with moderate noise (Gaussian noise with a standard deviation $\sigma = 0.03$ added to the measurements of the states);
- sampling time 1 and 250 samples with much noise (standard deviation $\sigma = 0.05$);
- sampling time 1/2 and 1000 samples, i.e. 50 trajectories, without noise;
- sampling time 1/2 and 1000 samples with moderate noise;
- sampling time 1/2 and 1000 samples with much noise.

We generally see a better performance from the symmetric mono-implicit Runge–Kutta methods (Midpoint, SRK4, SRK6) than those that only rely on one data point in the evaluation of $\Psi_{\Delta t}$. This seems to define a more important property than symplecticity; the symplectic PRK4 method performs consistently worse than the non-symplectic SRK4 method. The implicit midpoint method is the only method that is both symplectic and relies equally on x^n and x^{n+1}, and it does perform well despite being only a second-order method.

Note that while we train on the integration scheme in a way that is equivalent to performing one integration step, DiPietro et al. [16] perform several steps of the PRK4 integrator between data points to increase accuracy. We assume that this is a major reason for the superior performance over using the RK4 method reported in that paper, and also note that a major improvement in performance by taking many smaller integration steps will almost only happen if one trains on noise-free data.

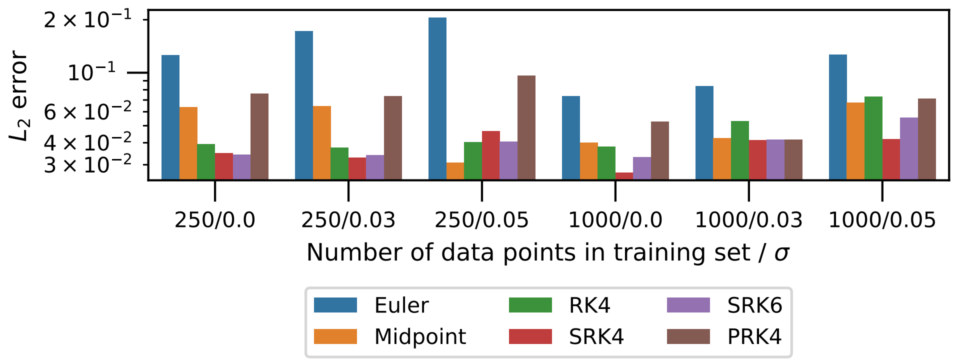

FIGURE 9. The mean L_2 error of PHSI models trained with the different integrators on the Hénon–Heiles problem. The error is on the predicted positions and momenta from $t = 0$ to $t = 10$ on 10 different random initial conditions.

B.2. **Pseudo-Hamiltonian system: Tanks and pipes.** We also test the performance of the different integrators in the hybrid model used to learn the tank system presented in Section 5.4. We do not test Yoshida's 4th order method on this problem, since the tank system is not a separable partitioned system, and so Yoshida's method is in this case an implicit method that would require a root-finding algorithm like Newton's method to be used in every iteration of the training process. We have not implemented this in our models, as it would be prohibitively expensive.

We train the models on trajectories from $t = 0$ to $t = 1$ with

- sampling time 1/50 and 7500 samples, i.e. 300 trajectories, without noise;
- sampling time 1/50 and 7500 samples with moderate noise (Gaussian noise with a standard deviation $\sigma = 0.03$ added to the measurements of the states);
- sampling time 1/50 and 7500 samples with much noise (standard deviation $\sigma = 0.05$);
- sampling time 1/100 and 30000 samples, i.e. 150 trajectories, without noise;
- sampling time 1/100 and 30000 samples with moderate noise;
- sampling time 1/100 and 30000 samples with much noise.

Figure 11 shows the L_2 error of all the state variables from the predictions obtained applying the methods on a test set of 10 different initial conditions sampled from independent uniform distributions $\mathcal{U}(-1, 1)$. The symmetric higher-order methods are more consistent in their performance than the lower-order methods and the explicit fourth-order Runge–Kutta method. As for the Hénon–Heiles system, we see no consistent improvement in going from fourth to sixth order, which supported the choice of using the SRK4 method throughout this paper. The symmetric methods particularly predict the tank volumes very well, more so than the pipe flows. An example is shown for the volume in the fourth tank, i.e. the one with a leak, in Figure 12.

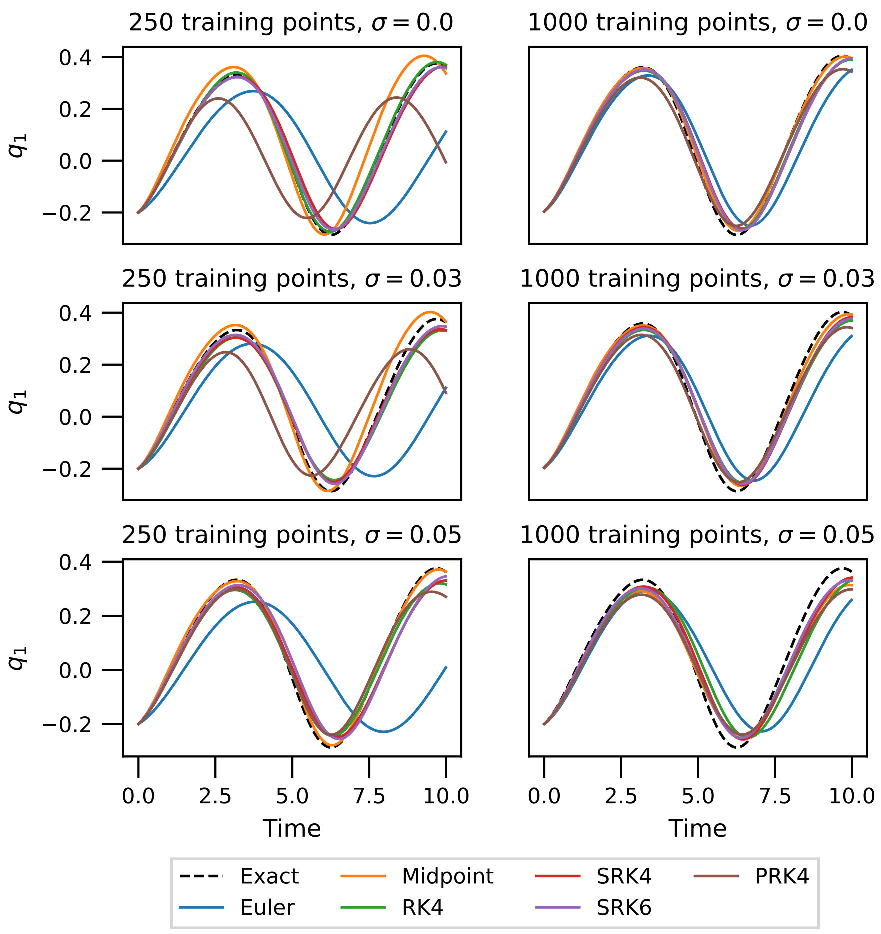

FIGURE 10. Prediction of q_1 by PHSI models trained with different integrators on the Hénon–Heiles problem. The initial condition is $q = (-0.2, 0.2)$, $p = (0.1, -0.2)$.

Table 7 shows how well the friction coefficients on the pipes are learned by the different integrators. We see here, even clearer than on the L_2 error, that the symmetric methods perform better than the explicit fourth-order Runge–Kutta method, particularly on noisy data.

Appendix C. A study of promoting sparsity through regularization and pruning. In sections 4.2 and 4.3, we argue that regularization and pruning will help the PHSI model promote sparsity in search of the true governing equations. It is also argued that since we know that the true governing equations are generally sparse in the function space, promoting sparsity during training will help the PHSI model better approximate the true system. To evaluate whether this claim has merit or not, we train PHSI models where the amounts of regularization and pruning vary and the rest of the hyperparameters are held constant. By the "amount" of pruning, we mean how often the pruning algorithm is employed during training, i.e. the value

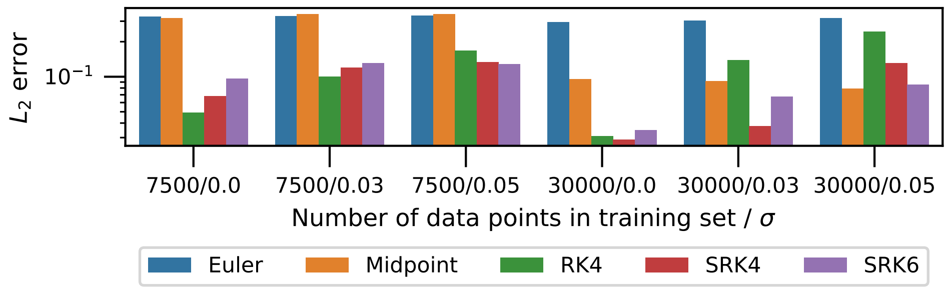

FIGURE 11. The mean L_2 error of hybrid PHSI models trained with the different integrators on the tank system. The error is of the predicted volume and flow in all tanks and pipes from $t = 0$ to $t = 1$ on 10 different random initial conditions.

TABLE 7. Mean and standard deviation of the predicted friction coefficients of the tank system, relative to the ground truth $R_p = (0.03, 0.03, 0.09, 0.03, 0.03)$ (i.e. so that 1 corresponds to the correct coefficient).

	no noise	$\sigma = 0.03$	$\sigma = 0.05$
	7500 training points		
Euler	22.75 ± 7.20	23.59 ± 7.04	25.27 ± 7.31
Midpoint	1.27 ± 0.04	1.16 ± 0.14	1.01 ± 0.11
RK4	1.04 ± 0.04	1.70 ± 0.24	1.95 ± 0.27
SRK4	1.16 ± 0.07	0.92 ± 0.13	0.88 ± 0.43
SRK6	0.95 ± 0.07	0.85 ± 0.19	0.94 ± 0.25
	30000 training points		
Euler	11.33 ± 3.31	12.31 ± 3.48	13.83 ± 3.91
Midpoint	1.23 ± 0.11	1.15 ± 0.16	1.16 ± 0.25
RK4	1.20 ± 0.06	1.97 ± 0.29	4.18 ± 0.70
SRK4	1.16 ± 0.04	1.11 ± 0.13	1.00 ± 0.23
SRK6	1.24 ± 0.06	1.21 ± 0.10	1.35 ± 0.32

P in Algorithm 1. The lower value P has, the more frequent the pruning. By the "amount" of regularization, we specifically mean the amount of l_1-regularization on the Hamiltonian $\hat{H}$, i.e. λ_H in (10). We evaluate the predictive performance of each trained model by comparing their "predictability score", i.e. their average error over simulated trajectories with random initialization compared to the true trajectories, where the trajectories are not from the training set. The tests are done for all of the four systems considered in the Section 5, and they are performed on noiseless data.

C.1. **Experiment 1: Hénon–Heiles system.** The Hénon–Heiles system is a purely Hamiltonian system, and this is assumed when training the PHSI model. The models are trained on noiseless, simulated data with $\lambda_H \in \{0, 0.05, 0.5\}$ and

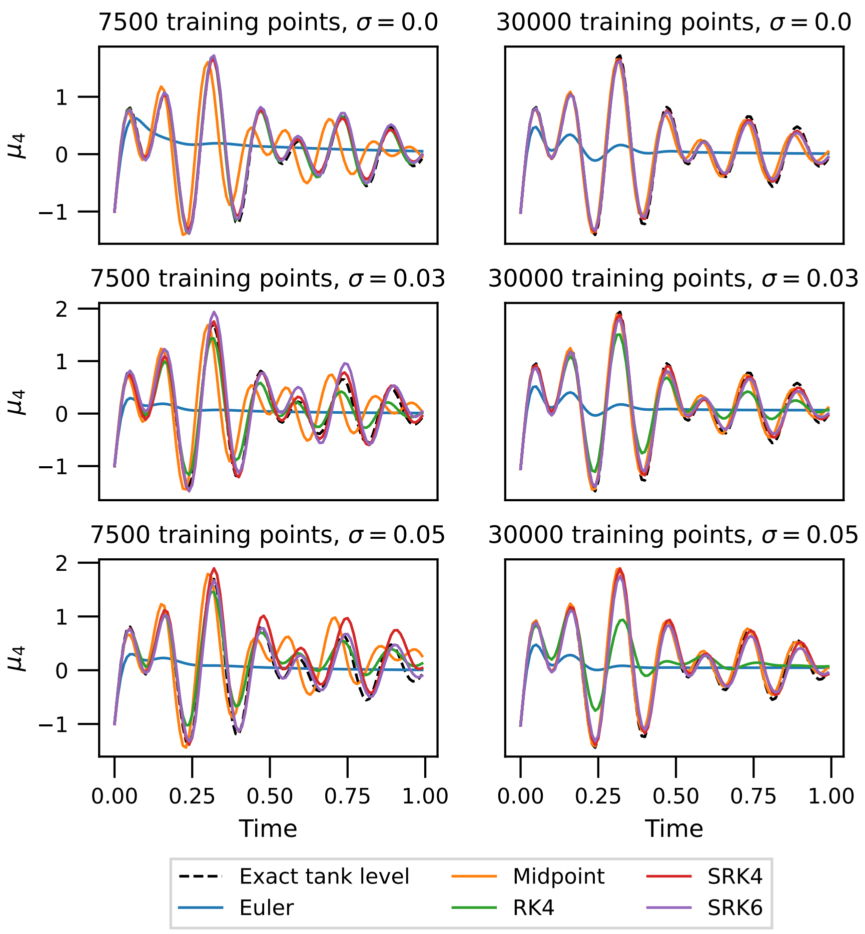

FIGURE 12. Volume of the fourth tank as predicted by hybrid PHSI models trained with different integrators on the tank system. The initial condition is $\phi^0 = (-1, -1, 0, \frac{1}{2}, -1)$, $\mu^0 = (1, 1, -\frac{1}{2}, -1)$.

$P \in \{1, 2, 4\}$, for 3 epochs. The predictive performances of models with different combinations of regularization and pruning are shown in Figure 13. The figure shows that for this particular system, the PHSI model trains best without l_1-regularization, and hence no regularization was used in the experiments in Section 5. The reason for this may lie in that the PHSI model learns the relatively simple Hénon–Heiles system in very few epochs of training. The pruning algorithm does not appear to affect the average L_2-error in this experiment. However, it excludes the correct terms, resulting in easier interpretability of the trained model. Figure 14 shows that although the scores in Figure 13 differ by several orders of magnitude, all the models achieve a relatively good predictive ability.

C.2. **Experiment 2: Nonlinear Schrödinger system.** The non-linear Schrödinger system is also purely Hamiltonian, and this is assumed when training the PHSI

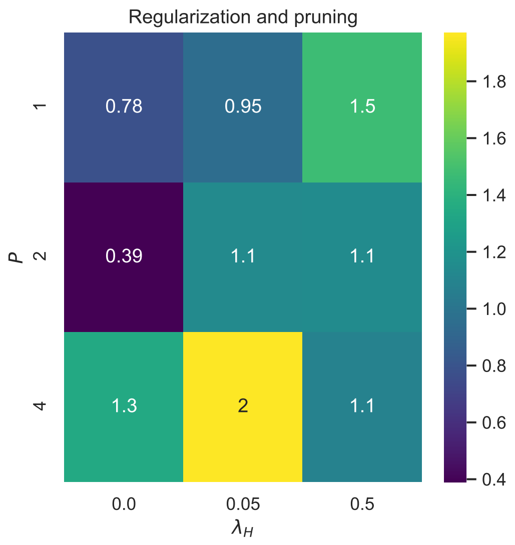

FIGURE 13. Average L_2-error over 30 simulated trajectories of the Hénon–Heiles system by PHSI models trained with different combinations of regularization parameter λ_H and pruning parameter P. The models are trained for 3 epochs, which means that when $P = 4$, there is no pruning.

model, in that we do not train damping coefficients or an external force. The models are trained with $\lambda_H \in \{0, 0.05, 0.5\}$ and $P \in \{1, 5, 10\}$, for 10 epochs. In Figure 15, we observe that the regularization improves the model performance drastically. The model with $P = 10$ and $\lambda_H = 0$ has an average L_2-error approximately 5 times that of the model with $P = 2$ and $\lambda_H = 0.5$, suggesting that the regularization and pruning work well together. This effect being so evident here and not for the Hénon–Heiles system may be due to the non-linear Schrödinger system being non-separable and of a higher polynomial order, making the true solution more sparse in the function space. Figure 16 illustrates the vastly superior performance of the models with more regularization and pruning over those with less.

C.3. **Experiment 3: Damped mass-spring system.** The damped mass-spring system described in Section 5.3 is a pseudo-Hamiltonian system. We have regularization on the Hamiltonian and the external forces, i.e. λ_H and λ_F in (10), but not on the damping coefficients. The models are trained for 80 epochs, with $\lambda_H \in \{0, 0.1, 0.5\}$ and $P \in \{20, 40, 80\}$. λ_F is set to 0.01. Figure 17 indicates that promoting sparsity helps the predictive performance of the PHSI model, especially through l_1-regularization. The superior predictive ability of the PHSI models that promote sparsity is also seen in the plotted simulated trajectories in Figure 18.

C.4. **Experiment 4: Tank system.** For the tank system considered in Section 5.4, we choose the search-spaces $\lambda_H \in \{0, 0.05, 0.5\}$ and $P \in \{10, 40, 80\}$, and train the models for 80 epochs. As shown in Figure 19, the regularization and pruning have a great impact on the performance of the model, and appear essential for achieving an accurate PHSI model. The model with $\lambda_H = 0$ and $P = 80$ has a

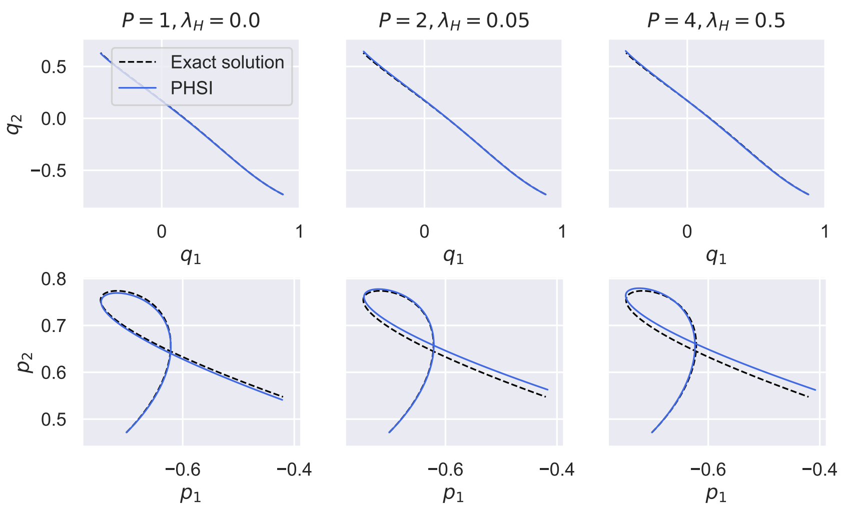

FIGURE 14. Simulated trajectories obtained for different PHSI models and the exact solution, for the nonlinear Hénon–Heiles system for three different combinations of regularization and pruning.

predictive score about 60 times that of the model with $\lambda_H = 0.5$ and $P = 10$. Figure 19 confirms that this assumption is true for the tank system.

In conclusion, promoting sparsity in the function space generally improves the accuracy of the PHSI model. Since promoting sparsity in the PHSI model through regularization and pruning improved results for three of the four models (it neither improved nor worsened the performance when learning the Hénon–Heiles system), the sparsity assumption made in sections 4.2 and 4.3 seems to be a good one. This again implies that the PHSI model is in general well fit for finding true governing equations. The experiments indicate that the regularization and pruning has more of an impact when learning systems of high dimensionality and polynomial order. If the PHSI model is to be applied to learning more complex and higher dimensional systems than what has been done in this thesis, promoting sparsity will presumably be vital for achieving an accurate model.

Appendix D. **A study of regularization: learning a mass-spring system.** When learning dynamical systems of complex structures, regularization is important not only to promote sparsity but to ensure the desired structure of the ODE. We exemplify this by studying a mass-spring system, without damping or external forces, and learning it through a pseudo-Hamiltonian structure (5). A simple-mass spring system can be described through the following ODE:

$$\begin{bmatrix}\dot{q}\\ \dot{p}\end{bmatrix} = \begin{bmatrix}\frac{p}{m}\\ -kq\end{bmatrix}. \tag{19}$$

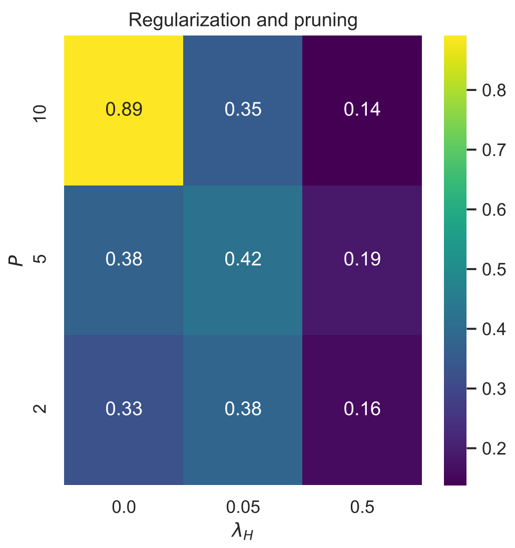

FIGURE 15. Average L_2-error over 30 simulated trajectories of the nonlinear Schrödinger system by PHSI models trained with different combinations of regularization parameter λ_H and pruning parameter P. The models are trained for 10 epochs, meaning when $P = 10$, the pruning only occurs once.

We set weight $m = 1$ and spring stiffness $k = 1$. The system then has the related Hamiltonian function

$$H(q,p) = \frac{1}{2}q^2 + \frac{1}{2}p^2, \tag{20}$$

which we can use to write the ODE as a Hamiltonian system with the canonical formulation

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{bmatrix}. \tag{21}$$

The model will be trained assuming a pseudo-Hamiltonian formulation (5) without damping, where the Hamiltonian and the external forces are trainable system identification models depending on the state only and the structure matrix is known:

$$\hat{g}_\theta(x) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \nabla \hat{H}_\theta(x) + \hat{F}_\theta(x). \tag{22}$$

As previously stated, the pseudo-Hamiltonian formulation is non-unique since a general ODE can be equivalently represented by different combinations of H and F. Since system identification models attempt not only to accurately predict future states but identify the governing equations, we are interested in separating $\hat{H}_\theta$ and $\hat{F}_\theta$ in a natural way. In this example, since there are no external forces present, the desired solution is $\hat{H}_\theta$ as given by (20) and $\hat{F}_\theta = 0$. We train system identification models with pseudo-Hamiltonian structures with different combinations of regularization on $\hat{H}_\theta$ and $\hat{F}_\theta$. Figure 21 shows trajectory plots of three randomly initiated models compared to the ground truth. Every model trained predicts $\hat{x}$ accurately, but only the models that regularize $\hat{F}_\theta$ are able to separate $\hat{H}_\theta$ and $\hat{F}_\theta$

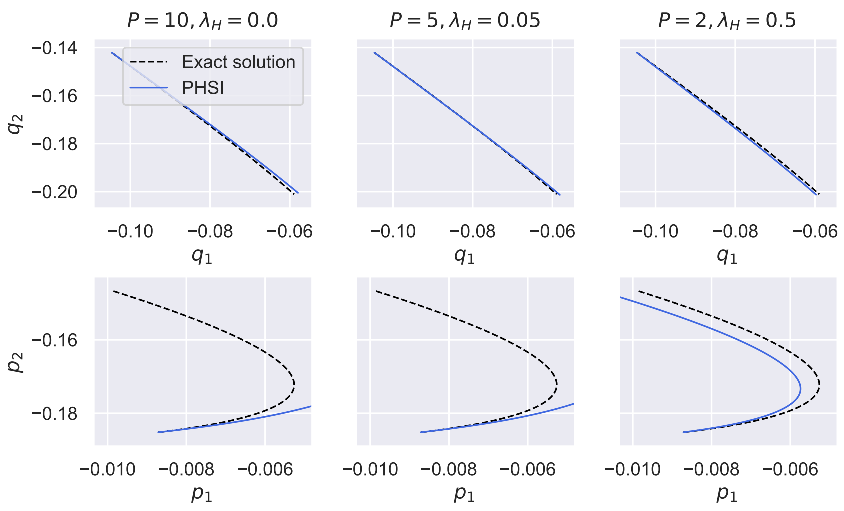

Figure 16. Simulated PHSI trajectories along with a simulation of the exact nonlinear Schödinger system, for three different combinations of regularization and pruning.

in the desired way. The models that regularize $\hat{H}_\theta$ learn the entire ODE through $\hat{F}_\theta$, and the models without regularization learn difficult-to-interpret combinations between $\hat{H}_\theta$ and $\hat{F}_\theta$ that are non-unique due to the random initialization of the model parameters. These results show how regularization affects the structure of a trained PHSI model. How we set the regularization parameters of a PHSI model reflects the prior assumptions we make about the system to be learned.

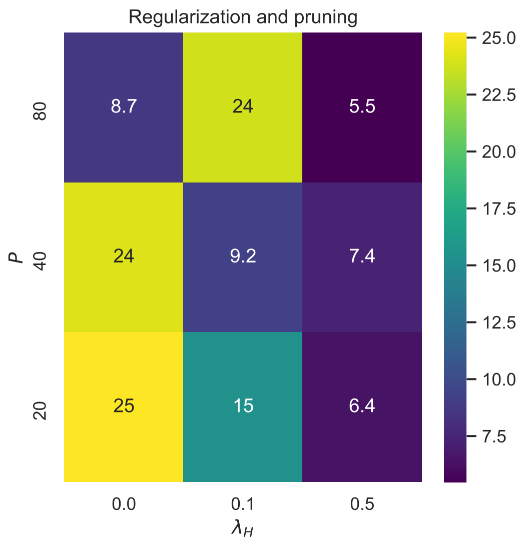

FIGURE 17. Average L_2-error over 30 simulated trajectories of the damped mass-spring system by PHSI models trained with different combinations of regularization-parameter λ_H and pruning-parameter P. The models are trained for 80 epochs, so that when $P = 80$, pruning is only done after the last epoch.

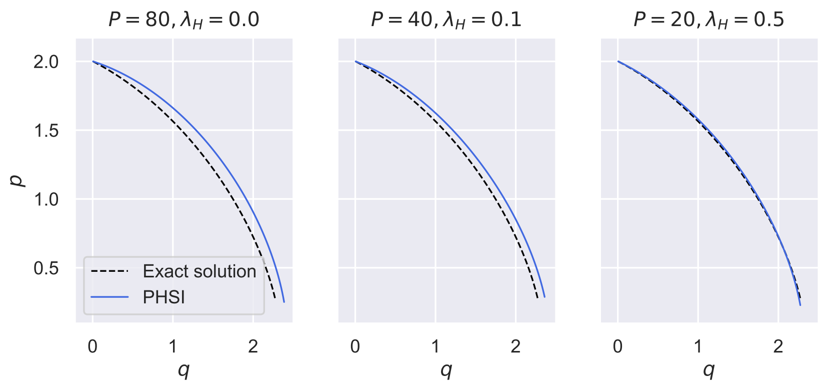

FIGURE 18. Simulated PHSI trajectories along with a simulation of the exact damped mass-spring system, for three different combinations of regularization and pruning.

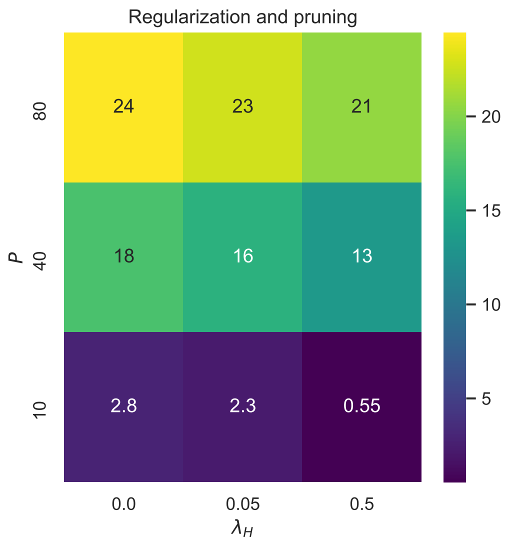

FIGURE 19. Average loss over simulated trajectories for PHSI models trained with different combinations of regularization-parameter λ_H and pruning-parameter P. The models are trained for 80 epochs, which means that when $P = 80$, pruning is only done after the last epoch.

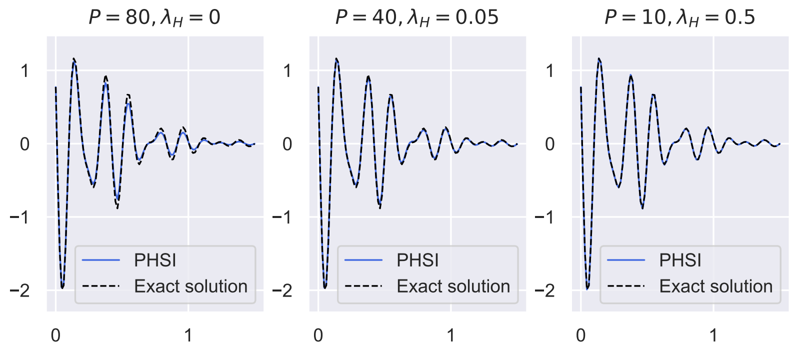

FIGURE 20. Simulated PHSI trajectories along with the exact solution of one of the leaking tanks in the connected tank system for three different combinations of regularization and pruning.

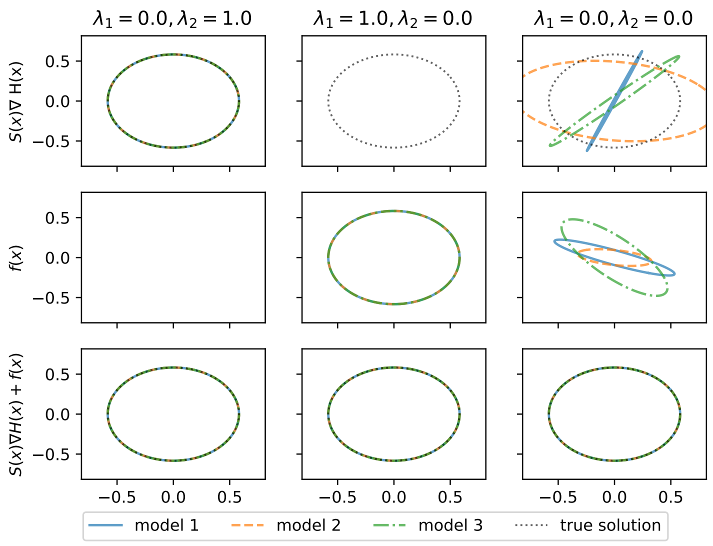

FIGURE 21. Phase plots of trajectories $x = (q, p)$ obtained from systems where $\hat{\dot{x}}$ is given by different combinations of $S(x)\nabla H$ and f. Each column is a different combination of regularization parameters, where λ_1 and λ_2 are the magnitudes of l_1 regularization on $\hat{H}_\theta$ and $\hat{F}_\theta$, respectively.