# DATA-EFFICIENT KERNEL METHODS FOR LEARNING HAMILTONIAN SYSTEMS

YASAMIN JALALIAN[1], MOSTAFA SAMIR[2], BOUMEDIENE HAMZI[1,3,5], PEYMAN TAVALLALI[4], AND HOUMAN OWHADI[1]

ABSTRACT. Hamiltonian dynamics describe a wide range of physical systems. As such, data-driven simulations of Hamiltonian systems are important for many scientific and engineering problems. In this work, we propose kernel-based methods for identifying and forecasting Hamiltonian systems directly from data. We present two approaches: a two-step method that reconstructs trajectories before learning the Hamiltonian, and a one-step method that jointly infers both. Across several benchmark systems, including mass-spring dynamics, a nonlinear pendulum, and the Hénon-Heiles system, we demonstrate that our framework achieves accurate, data-efficient predictions and outperforms two-step kernel-based baselines, particularly in scarce-data regimes, while preserving the conservation properties of Hamiltonian dynamics. Moreover, our methodology provides theoretical *a priori* error estimates, ensuring reliability of the learned models. We also provide a more general, problem-agnostic numerical framework [1] that goes beyond Hamiltonian systems and can be used for data-driven learning of arbitrary dynamical systems.

## 1. Introduction

Dynamical systems play a fundamental role in describing natural phenomena. Two essential aspects of their study are (i) *system identification*, which involves determining parametric or nonparametric models for the system's evolution, and (ii) *prediction or forecasting*, which focuses on projecting the system's state forward in time. With the advent of machine learning and its growing role in inferring data through mathematical models, known as physics-informed machine learning, data-driven methods have become increasingly important for treating both tasks [11, 39, 13, 35, 55, 24, 23, 15, 54, 51, 1, 26, 57].

Among the various learning approaches, reproducing kernel Hilbert spaces (RKHS) [19, 38] have established solid mathematical foundations for the analysis of dynamical systems [3, 7, 8, 9, 43, 34, 47, 44, 32, 46, 42, 41, 29, 22, 27, 28, 30, 35, 6, 31], surrogate modeling [56], and neural networks [58]. Compared to other learning approaches, kernel-based methods offer advantages in interpretability, theoretical analysis, numerical implementation and regularization, and uncertainty quantification (UQ), while also providing guaranteed convergence along with *a priori* error estimates [36, 4, 16, 49, 45, 52].

A particularly interesting class of dynamical systems is Hamiltonian dynamics. These systems naturally arise in describing the evolution of a wide range of physical systems. They are especially important as their structure preserves fundamental quantities like energy and follows specific geometric principles. The problem of learning Hamiltonian dynamical systems from data has been treated recently using different approaches [5, 12, 14, 18, 33, 48, 60], including neural networks

[1] DEPARTMENT OF COMPUTING AND MATHEMATICAL SCIENCES, CALTECH, CA, USA.
[2] BEYOND LIMITS, CA, USA.
[3] THE ALAN TURING INSTITUTE, LONDON, UK.
[5] PARTS OF THIS WORK WERE CARRIED OUT WHILE THE AUTHOR WAS A RESIDENT SCHOLAR AT THE ISAAC NEWTON INSTITUTE FOR MATHEMATICAL SCIENCES, CAMBRIDGE, UK.
[4] JPL, CALTECH, CA, USA.
*E-mail addresses*: yjalalia@caltech.edu, mibrahim@beyond.ai, boumediene.hamzi@gmail.com, tavallali@gmail.com, owhadi@caltech.edu.
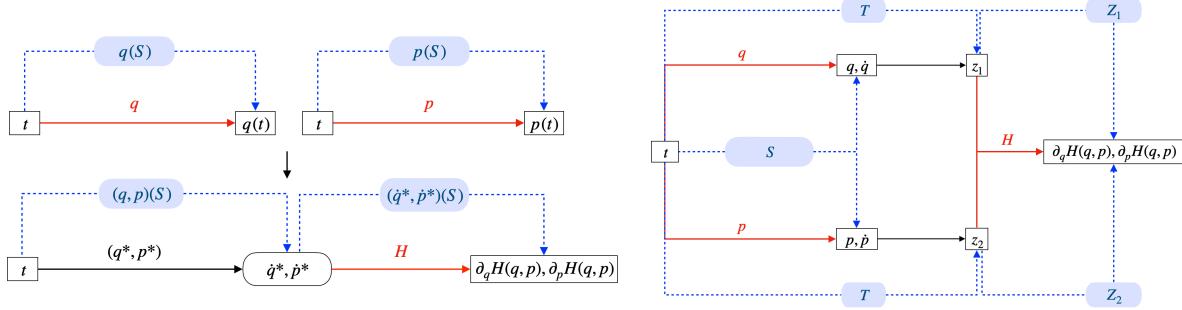
FIGURE 1. Computational graph of the 2-step method (left) - Computational graph of the 1-step method (right).

designed to preserve symplectic structure [17, 18, 37, 25, 61, 62], Gaussian processes combined with variational integrators for structure-preserving modeling [59], and methods for discovering governing equations and coordinates from data [40, 14].

More concretely, consider the Hamiltonian system

$$\begin{cases} \dot{q} = \partial_p H, \\ \dot{p} = -\partial_q H, \end{cases}$$

where $q(t) \in \mathbb{R}^m$ and $p(t) \in \mathbb{R}^m$ represent the generalized position and the generalized momentum at time $t$, respectively, and $H(q, p)$ is the Hamiltonian. Given discretized time series data on the interval $[0, T_{\text{final}}]$ for trajectories $q$ and $p$ and knowing that they come from an unknown Hamiltonian system, the problem of system identification corresponds to identifying $q, p$ and $H$ over the entire domain $[0, T_{\text{final}}]$, whereas the problem of forecasting corresponds to predicting values of $q$ and $p$ after time $T_{\text{final}}$ using the identified Hamiltonian.

In this work, we address both of these problems by using a similar methodology to the one of Kernel Equation Learning (KEqL) [36] which addresses problems in PDE learning. In particular, we propose two kernel-based methods, one that first aims at learning trajectories $q, p$ and then learns Hamiltonian $H$ and another method that simultaneously learns $q, p$ and $H$. A visual scheme of the architecture is given in figure 1. Inspired by the Computational Graph Completion (CGC) framework [52], we model our problem as a computational graph problem by representing state variables as vertices, and functional dependencies between variables as directed edges. We represent unknown functions as red edges, and data as dotted edges. Then solving our problem is equivalent to completing the graph by approximating the unknown functions with the minimizers of optimal recovery problems over suitably-chosen reproducing kernel Hilbert spaces (RKHSs). This methodology is equivalent to replacing the unknown functions of the graph with Gaussian Processes and computing their MAP estimators given constraints imposed by the data and the structure of the graph. The RKHS perspective inherits the convergence and error estimate guarantees for kernel methods. Moreover, the GP perspective allows us to access the whole posterior distribution of point-wise approximations and employ them for UQ estimates.

We demonstrate the efficiency of our simultaneous learning method compared to the two-step kernel-based approach, particularly in a scarce-data regime, across various physical problems. Our proposed framework is mathematically interpretable, and the architecture inherently results in a system with conserved energy. Additionally, we provide theoretical error bounds to assess the reliability of our predictions. Moreover, the numerical framework [1] that we provide with this work

---

[1]The framework can be found at: https://github.com/Mostafa-Samir/CGC.

is a problem-agnostic general CGC implementation that goes beyond solving the problem at hand and can be used for inferring unknown variables in any arbitrary physical system from data.

The paper is outlined as follows. In section 2, we begin by introducing the problem formally and proceed to describing the two suggested methods for its solution. In section 3, we prove *a priori* error rates for the 1-step methods. Section 4 is dedicated to discussion on the details of the implementation for both methods. In section 5, we demonstrate numerical results on several benchmark systems and discuss how they compare with each other in different data regimes. Finally, section 6 is dedicated to concluding remarks and future directions. A preliminary version of this paper was presented at *Differential Equations for Data Science 2023 (DEDS2023)* (video, conference site).

## 2. Problem Statement & Methodology

### 2.1. Problem Statement. We start by formally introducing the problem and setting up notation. Let $H : \mathbb{R}^{2m} \to \mathbb{R}$ be an unknown Hamiltonian defining a dynamical system via the equations

$$\begin{cases} \dot{q} = \partial_p H(q, p), \\ \dot{p} = -\partial_q H(q, p), \end{cases}$$

where $q : \mathbb{R} \to \mathbb{R}^m$, $p : \mathbb{R} \to \mathbb{R}^m$, denote the generalized position and momentum respectively. At time $t \in \mathbb{R}_+$, we define the state of the full system as

$$y(t) := \big(q(t), p(t)\big)^\top \in \mathbb{R}^{2m},$$

so that its evolution can be compactly written as

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \begin{pmatrix} \partial_p H(q(t), p(t)) \\ -\partial_q H(q(t), p(t)) \end{pmatrix}.$$

Let $T_{\mathrm{col}} := \{t_i\}_{i=1}^N$ be a discretization of the interval $[0, T_{\mathrm{final}}]$, and let $\{y(t_i)\}_{i=1}^N$ constitute our full set of *collocation points*. To simulate partial observability within the collocation points, we introduce a *sparsity factor* $\alpha \in (0, 1]$. This determines the proportion of collocation points that are *not* observed. Specifically, we define

$$N_{\mathrm{obs}} := \lfloor (1 - \alpha)N \rfloor,$$

and uniformly at random select $N_{\mathrm{obs}}$ of the $N$ time points. We define

$$S_{\mathrm{obs}} := \{s_j\}_{j=1}^{N_{\mathrm{obs}}} \subset T_{\mathrm{col}}$$

and furthermore the set $\{y(s_j) | s_j \in S_{\mathrm{obs}}\}$ as our *observation points*.

For an example system, in Figure 2, we plot the phase-space trajectories for each mass component, showing both the full trajectories on the collocation points (in black) and the observed points (in red).

Our goal is to recover $q, p, H$ from these observation and collocation points and to forecast $q, p$ in time. Here we focus on one-shot learning and consider a setting where data comes from the partial observation of a single trajectory. We remark however that our proposed methodology can be easily extended to a setting where data is obtained from multiple trajectories.

### 2.2. The Proposed Method. In the following, we outline our proposed methodology for learning and forecasting the Hamiltonian system based on the theory of reproducing kernel Hilbert spaces, closely following the methodology introduced in [36] for equation learning. To begin, we represent the kernels we are going to use throughout the paper for approximating $q, p, H$. Let $\Gamma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and $\Sigma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be positive-definite kernels with respective RKHS(s) $\mathcal{H}_\Gamma$ and $\mathcal{H}_\Sigma$, and let $\Psi : \mathbb{R}^{2m} \times \mathbb{R}^{2m} \to \mathbb{R}$ be a positive-definite kernel with RKHS $\mathcal{H}_\Psi$. For precise definitions of the kernels used in numerical experiments, see subsection 4.5.
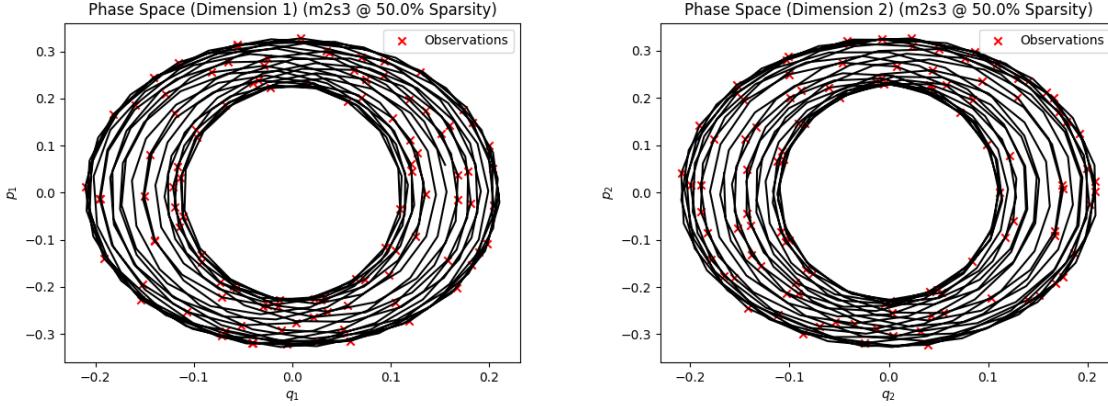
FIGURE 2. Phase space trajectories of the two-mass-three-spring system with sparsity factor 0.5.

2.2.1. *2-step method: first learn $q, p$, then learn $H$.* The first method we study is the classic kernel interpolation method. At the first step, we interpolate the functions $q$ and $p$. Formally, we denote the observation vector and the collocation vector as

$$S := \left(s_1, \ldots, s_{N_{\mathrm{obs}}}\right)^\top \in \mathbb{R}^{N_{\mathrm{obs}}}, \quad \text{for } s_j \in S_{\mathrm{obs}},$$

$$T := \left(t_1, \ldots, t_N\right)^\top \in \mathbb{R}^N, \qquad \text{for } t_i \in T_{\mathrm{col}},$$

and we approximate each function $q$ and $p$ by finding the interpolant with the minimum RKHS norm. That is, we solve

$$q^\star := \underset{\widehat{q} \in \mathcal{H}_\Gamma}{\arg\min} \quad \left\|\widehat{q}\right\|_\Gamma \quad \text{subject to} \quad \widehat{q}(S) = q(S), \tag{1}$$

and

$$p^\star := \underset{\widehat{p} \in \mathcal{H}_\Sigma}{\arg\min} \quad \left\|\widehat{p}\right\|_\Sigma \quad \text{subject to} \quad \widehat{p}(S) = p(S). \tag{2}$$

At the second step, using the interpolants of the first step, we find the best approximation for the function $H$. In particular, we explicitly compute the derivatives of the approximants $q^\star, p^\star$ from the first step, and we solve

$$H^\star := \underset{\widehat{H} \in \mathcal{H}_\Psi}{\arg\min} \quad \|\widehat{H}\|_\Psi^2 + \frac{1}{\lambda_1}\|\partial_p \widehat{H}((q,p)(T)) - \dot{q}^\star(T)\|_2^2 + \frac{1}{\lambda_2}\|\partial_q \widehat{H}((q,p)(T)) + \dot{p}^\star(T)\|_2^2, \tag{3}$$

where $\lambda_1, \lambda_2 > 0$ are regularization parameters. It is well known that the optimization problems above are well-posed and admit unique solutions $q^\star, p^\star, H^\star$; more details are given in section 4.

2.2.2. *1-step method: simultaneously learn $q, p$ and $H$.* In the second method, we estimate $q, p, H$ by a single joint optimization problem, enforcing the constraint that $(q, p)$ solve a system of ODEs defined by $H$. For ease of notation, we keep the same notation used for the 2-step method but

make explicit which method is being used. We solve

$$(q^\star, p^\star, H^\star) := \underset{\widehat{q} \in \mathcal{H}_\Gamma, \widehat{p} \in \mathcal{H}_\Sigma, \widehat{H} \in \mathcal{H}_\Psi}{\arg\min} \quad \|\widehat{H}\|_\Psi^2 + \frac{1}{\lambda_1}\|\widehat{q}\|_\Gamma^2 + \frac{1}{\lambda_2}\|\widehat{p}\|_\Sigma^2$$

$$\text{subject to} \quad \begin{cases} \widehat{q}(S) = q(S), \\ \widehat{p}(S) = p(S), \\ \dot{\widehat{q}}(T) = \partial_p \widehat{H}((q,p)(T)), \\ \dot{\widehat{p}}(T) = -\partial_q \widehat{H}((q,p)(T)), \end{cases} \tag{4}$$

where the first two constraints are the observation constraints and the last two are the constraints enforcing the physics of the problem on collocation points. The optimization problem 4 is well-posed and section 4 shows how we compute its minimizers.

**2.3. Comparison of the 2-step and 1-step methods.** In the 2-step method, we discard the physics constraints in the first step. Then, in the second step, we compute derivatives of the approximants from the first step to create values for the optimization problem of the second step. In particular, in a scarce-data regime, the derivative of the approximant may not provide an accurate approximation for the derivative of the true unknown functions. However, the 1-step method surmounts these obstacles by taking into account all information from data and the physics of the problem simultaneously. Therefore, from a methodological perspective, the 1-step method is more advantageous to use in a scarce-data regime. This argument is numerically validated on various examples in section 5.

## 3. Theoretical Analysis

In this section, we present *a priori* error bounds for the 1-step method introduced in section 2.2.2. These bounds quantify the approximation error of the learned trajectories and Hamiltonian in terms of the smoothness of the true functions and the fill distance of the sampled collocation points. The analysis builds on classical results in the approximation theory of RKHS(s) [64, 63, 50, 21], and adapts them to the context of Hamiltonian systems with coupled derivative constraints. For an expanded treatment of both 1-step and 2-step convergence theory for a PDE learning setup, see [36, Supplementary Information B]. We will furthermore refer to some lemmas and theorems proved in [36]; for convenience, we have restated them in appendix A.

For this section, we rewrite the optimization problem 4 in the following equivalent form

$$(y_N^\star, H_N^\star) := \underset{\widehat{y}, \widehat{H}}{\arg\min} \|\widehat{H}\|_\Psi^2 + \frac{1}{\lambda_1}\|\widehat{y}_1\|_\Gamma^2 + \frac{1}{\lambda_2}\|\widehat{y}_2\|_\Sigma^2$$

$$\text{subject to} \quad \begin{cases} \widehat{y}(S) = y(S), \\ \dot{\widehat{y}}(T) = J\nabla\widehat{H}(y(T)), \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}. \end{cases} \tag{5}$$

Note that, in contrast to the previous notation, in this section, we have made the dependence of the estimates on $N$ explicit by referring to them as $Y_N^\star, H_N^\star$. Our goal here is to get *a priori* error rates for these reconstructed functions $y_N^\star, H_N^\star$. Notice however that the equality constraints of optimization problem 5 are only on $\nabla H$ and not directly on $H$. Thus, we can only get rates for the recovery of $\nabla H$ which is equivalent to the recovery of $H$ up to a constant.

To establish our error analysis, we require certain regularity conditions and fill-distance assumptions for $q, p$, and $H$. We state these assumptions below. Without loss of generality, we assume that $\lambda_1, \lambda_2$, the weights in the optimal recovery problem 4, are both equal to one as they only affect the constant and not the rates.

**Assumption 3.1 (Assumptions on $q, p$).** *Assume that the following holds:*

(i) $\Gamma, \Sigma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ are continuous, positive-definite kernels with associated RKHS(s) $\mathcal{H}_\Gamma, \mathcal{H}_\Sigma$.

(ii) $\mathcal{H}_\Gamma$ is continuously embedded in the Sobolev space $H^\gamma([0, T_{\text{final}}]; \mathbb{R}^m)$ for some $\gamma > 0$ and $q \in H^\gamma([0, T_{\text{final}}]; \mathbb{R}^m)$;

(iii) $\mathcal{H}_\Sigma$ is continuously embedded in the Sobolev space $H^\sigma([0, T_{\text{final}}]; \mathbb{R}^m)$ for some $\sigma > 0$ and $p \in H^\sigma([0, T_{\text{final}}]; \mathbb{R}^m)$;

(iv) the observation times $S_{\text{obs}} \subset [0, T_{\text{final}}]$ are sampled with fill distance $\rho_N$ defined as

$$\rho_N := \sup_{t \in [0, T_{\text{final}}]} \inf_{t' \in S_{\text{obs}}} |t - t'|.$$

**Assumption 3.2 (Assumptions on $H$).** *Assume that the following holds:*

(i) $\Psi : \mathbb{R}^{2m} \times \mathbb{R}^{2m} \to \mathbb{R}$ is a continuous, positive-definite kernel with associated RKHS $\mathcal{H}_\Psi$.

(ii) $\mathcal{H}_\Psi$ is continuously embedded in the Sobolev space $H^\eta(\Omega; \mathbb{R})$ for some $\eta > m + 1$ and some $\Omega \subset \mathbb{R}^{2m}$ a compact domain containing the collocation trajectory image $\mathcal{Z} := \{y(t) : t \in T_{\text{col}}\}$, and $H \in H^\eta(\Omega; \mathbb{R})$;

(iii) the collocation trajectory image points $\mathcal{Z}$ are sampled with fill distance $\varrho_N$ defined as

$$\varrho_N := \sup_{x \in \Omega} \inf_{x' \in \mathcal{Z}} \|x - x'\|_2.$$

**Theorem 3.1 (A priori error bounds for the 1-step method).** *Assume that assumptions 3.1 and 3.2 hold. Let the functions $q_N^\star \in \mathcal{H}_\Gamma$, $p_N^\star \in \mathcal{H}_\Sigma$, and $H_N^\star \in \mathcal{H}_\Psi$ be the minimizers of the 1-step optimization problem 5. Then, there exist constants $\rho_0, \varrho_0 \in (0, 1)$ so that whenever $\rho_N < \rho_0$ and $\varrho_N < \varrho_0$ it holds that*

(a) *if $q \in \mathcal{H}_\Gamma$ and $p \in \mathcal{H}_\Sigma$, then for any smoothness indices $0 \leqslant \gamma' < \gamma$, $0 \leqslant \sigma' < \sigma$, there exist constants $C_q, C_p > 0$ such that*

$$\|q_N^\star - q\|_{H^{\gamma'}}^2 \leqslant C_q \, \rho_N^{2(\gamma - \gamma')} \left( \|H\|_\Psi^2 + \|q\|_\Gamma^2 + \|p\|_\Sigma^2 \right),$$

$$\|p_N^\star - p\|_{H^{\sigma'}}^2 \leqslant C_p \, \rho_N^{2(\sigma - \sigma')} \left( \|H\|_\Psi^2 + \|q\|_\Gamma^2 + \|p\|_\Sigma^2 \right);$$

(b) *if $q \in \mathcal{H}_\Gamma^2$, $p \in \mathcal{H}_\Sigma^2$, and $\sigma, \gamma > 3/2$, then for any smoothness indices $0 \leqslant \eta' < \eta$, $3/2 < \gamma' < \gamma$, $3/2 < \sigma' < \sigma$, there exists a constant $C_H > 0$ such that*

$$\|\nabla H - \nabla H_N^\star\|_{L^\infty(\Omega)}^2 \leqslant C_H \left( \varrho_N^{2(\eta - 1 - m)} + \rho_N^{2(\gamma - \gamma')} + \rho_N^{2(\sigma - \sigma')} \right) \left( \|H\|_\Psi^2 + \|q\|_{\Gamma^2}^2 + \|p\|_{\Sigma^2}^2 \right),$$

*where the nested RKHSs $\mathcal{H}_\Gamma^2, \mathcal{H}_\Sigma^2$ and their corresponding norms are defined in appendix A.*

*Proof.* (a) Observe that $(q_N^\star - q)(S_{\text{obs}}) = 0$ as $q_N^\star$ is interpolating $q$ passing through $S_{\text{obs}}$ and so we can apply the noiseless Sobolev sampling inequality A.2(a). Indeed, directly applying that result followed by the continuous embedding assumption 3.1(ii) we obtain that for $\rho_N < \rho_0$,

$$\|q_N^\star - q\|_{H^{\gamma'}} \leqslant C \, \rho_N^{\gamma - \gamma'} \|q_N^\star - q\|_{H^\gamma} \leqslant \tilde{C} \, \rho_N^{\gamma - \gamma'} \|q_N^\star - q\|_\Gamma.$$

Using the triangle inequality, the identity $(a+b)^2 \leqslant 2(a^2 + b^2)$, and $\min a + \min b \leqslant \min(a+b)$ for $q_N^\star$, we can write

$$\|q_N^\star - q\|_{H^{\gamma'}}^2 \leqslant 2 \tilde{C}^2 \, \rho_N^{2(\gamma - \gamma')} \left( \|q_N^\star\|_\Gamma^2 + \|q\|_\Gamma^2 \right) \leqslant C_q \, \rho_N^{2(\gamma - \gamma')} \left( \|H\|_\Psi^2 + \|q\|_\Gamma^2 + \|p\|_\Sigma^2 \right). \tag{6}$$

The proof for $p$ is done similarly.

(b) Let us break down the proof into several parts.

**Finding a uniform bound for $H_N^*$:** For $q$, define the optimal interpolant

$$\bar{q}_N := \arg\min_{r \in \Gamma} \|r\| \quad \text{s.t.} \quad r(S) = q(S).$$

Representer theorem gives $\bar{q}_N = \Gamma(S, \cdot)^\top \Gamma(S, S)^{-1} q(S)$ and a direct calculation using the reproducing property implies that $\langle q - \bar{q}_N, \bar{q}_N \rangle_\Gamma = 0$ which in turn gives the identity

$$\|q - \bar{q}_N\|_\Gamma^2 = \|q\|_\Gamma^2 - \|\bar{q}_N\|_\Gamma^2.$$

Moreover, by $\min a + \min b \leqslant \min(a + b)$, we have the bound $\|\bar{q}_N\|_\Gamma \leqslant \|q_N^\star\|_\Gamma$. Thus we obtain the lower bound

$$\|q_N^\star\|_\Gamma^2 \geqslant \|q\|_\Gamma^2 - \|q - \bar{q}_N\|_\Gamma^2. \tag{7}$$

On the other hand, thanks to the assumption that $q \in \mathcal{H}_\Gamma^2$, applying lemma A.1 then gives the bound

$$\|q - \bar{q}_N\|_\Gamma^2 \leqslant \|q - \bar{q}_N\|_{L^2([0,T_{\text{final}}])} \|q\|_{\Gamma^2},$$

and since $(\bar{q}_N - q)(S_{\text{obs}}) = 0$, we apply the noiseless Sobolev sampling inequality A.2(a) to the bound above to further get the bound

$$\|q - \bar{q}_N\|_\Gamma^2 \leqslant \bar{C} \rho_N^\gamma \|q\|_\Gamma \|q\|_{\Gamma^2}.$$

Putting 7 and the bound above together, and following a similar procedure for $p$, we get the bounds

$$\begin{aligned} \|q\|_\Gamma^2 - \|q_N^\star\|_\Gamma^2 &\leqslant \bar{C} \rho_N^\gamma \|q\|_\Gamma \|q\|_{\Gamma^2}, \\ \|p\|_\Sigma^2 - \|p_N^\star\|_\Sigma^2 &\leqslant \bar{C} \rho_N^\sigma \|p\|_\Sigma \|p\|_{\Sigma^2}, \end{aligned} \tag{8}$$

where by abuse of notation, we redefine $\bar{C}$ as the maximum of the constants for $q$ and $p$. Finally, the optimality of $q_N^\star, p_N^\star, H_N^\star$ gives

$$\|H_N^\star\|_\Psi^2 \leqslant \|H\|_\Psi^2 + \left[\|q\|_\Gamma^2 - \|q_N^\star\|_\Gamma^2\right] + \left[\|p\|_\Sigma^2 - \|p_N^\star\|_\Sigma^2\right],$$

and putting together 8 with the bound above gives

$$\|H_N^\star\|_\Psi^2 \leqslant \|H\|_\Psi^2 + \bar{C}\left(\rho_N^\gamma \|q\|_\Gamma \|q\|_{\Gamma^2} + \rho_N^\sigma \|p\|_\Sigma \|p\|_{\Sigma^2}\right),$$

which by using the embeddings $\mathcal{H}_\Sigma^2 \subset \mathcal{H}_\Sigma, \mathcal{H}_\Gamma^2 \subset \mathcal{H}_\Gamma$, the assumptions that $\rho_N < 1$ and $\gamma, \sigma > 0$, and bounding all the constants by their maximum value and denoting it as $\bar{C}_{\text{emb}}$ further gives the bound

$$\|H_N^\star\|_\Psi^2 \leqslant \bar{C}_{\text{emb}} \left(\|H\|_\Psi^2 + \|q\|_{\Gamma^2}^2 + \|p\|_{\Sigma^2}^2\right). \tag{9}$$

Thus, we have an *a priori* bound on the RKHS norm of the reconstruction $H_N^\star$ that depends only on the true functions and fill distance.

**Controlling the error between $\nabla H$ and $\nabla H_N^\star$:** Fix $s \in S_{\text{obs}}$. From equality constraints in 5, and from the Hamiltonian equations for true dynamics, we know that

$$\dot{y}(s) = J\nabla H(y(s)), \qquad \dot{y}_N^\star(s) = J\nabla H_N^\star\big(y(s)\big),$$

where $\|J\|_2 = 1$, so

$$\left\|\nabla H(y(s)) - \nabla H_N^\star(y(s))\right\|_2 \leqslant \left\|\dot{y}(s) - \dot{y}_N^\star(s)\right\|_2 \leqslant \left\|\dot{y} - \dot{y}_N^\star\right\|_{L^\infty([0,T_{\text{final}}])}. \tag{10}$$

Now, note that by triangle inequality on the two-norms and taking sup, we have

$$\|\dot{y} - \dot{y}_N^\star\|_{L^\infty([0,T_{\text{final}}])} \leqslant \|\dot{q} - \dot{q}_N^\star\|_{L^\infty([0,T_{\text{final}}])} + \|\dot{p} - \dot{p}_N^\star\|_{L^\infty([0,T_{\text{final}}])}. \tag{11}$$

For $\gamma', \sigma' > 3/2$, by Sobolev embedding theorem (with $\alpha, d = 1$ in theorem A.1),

$$H^{\gamma'}([0, T_{\text{final}}]), H^{\sigma'}([0, T_{\text{final}}]) \hookrightarrow C^1([0, T_{\text{final}}]).$$

In particular, there exist $C_{\text{emb,q}}, C_{\text{emb,p}} > 0$ such that

$$\begin{aligned} \|\dot{q} - \dot{q}_N^\star\|_{L^\infty([0,T_{\text{final}}])} &\leqslant \|q - q_N^\star\|_{C^1([0,T_{\text{final}}])} \leqslant C_{\text{emb,q}} \|q - q_N^\star\|_{H^{\gamma'}([0,T_{\text{final}}])}, \\ \|\dot{p} - \dot{p}_N^\star\|_{L^\infty([0,T_{\text{final}}])} &\leqslant \|p - p_N^\star\|_{C^1([0,T_{\text{final}}])} \leqslant C_{\text{emb,p}} \|p - p_N^\star\|_{H^{\sigma'}([0,T_{\text{final}}])}. \end{aligned} \tag{12}$$

Denote $C_{\text{emb}} = \max(C_{\text{emb,q}}, C_{\text{emb,p}})$. Putting equations 6, 11 and 12 together and again using $(a + b)^2 \leqslant 2(a^2 + b^2)$, we get

$$\|\dot{y} - \dot{y}_N^\star\|_{L^\infty([0,T_{\text{final}}])}^2 \leqslant 2\, C_{\text{emb}}^2 \left( C_q\, \rho_N^{2(\gamma - \gamma')} + C_p\, \rho_N^{2(\sigma - \sigma')} \right) \left( \|H\|_\Psi^2 + \|q\|_\Gamma^2 + \|p\|_\Sigma^2 \right).$$

Putting together equations 10 with the bound above, we get that for $s \in S_{\text{obs}}$,

$$\left\|(\nabla H - \nabla H_N^\star)\big(y(s)\big)\right\|_2^2 \leqslant 2\, C_{\text{emb}}^2 \left( C_q\, \rho_N^{2(\gamma - \gamma')} + C_p\, \rho_N^{2(\sigma - \sigma')} \right) \left( \|H\|_\Psi^2 + \|q\|_\Gamma^2 + \|p\|_\Sigma^2 \right). \qquad (13)$$

We have thus shown that $H$ and $H_N^\star$ are close on the discrete set $S_{\text{obs}}$.

**Discrete to continuous:** Assumption 3.2(ii) implies $\nabla H \in H^{\eta - 1}(\Omega; \mathbb{R}^{2m})$ and by the noisy Sobolev sampling inequality A.2(b), under the assumption that $\varrho_N \leqslant \varrho_0$, we get

$$\|\nabla H - \nabla H_N^\star\|_{L^\infty(\Omega)} \leqslant C^\star \varrho_N^{\eta - 1 - m} \|\nabla H - \nabla H_N^\star\|_{H^{\eta-1}(\Omega)} + 2\|(\nabla H - \nabla H_N^\star)|_{S_{\text{obs}}}\|_\infty. \qquad (14)$$

Using the fact that $\nabla : H^\eta(\Omega) \to H^{\eta-1}(\Omega; \mathbb{R}^{2m})$ is a bounded linear operator, the continuous embedding assumption 3.2(ii) and triangle inequality, we can further write

$$\|\nabla H - \nabla H_N^\star\|_{H^{\eta-1}(\Omega)} \leqslant C_{\Omega,\eta}\|H - H_N^\star\|_{H^\eta(\Omega)} \leqslant \widehat{C}\left( \|H\|_\Psi + \|H_N^\star\|_\Psi \right),$$

which together with 9 and $(a + b)^2 < 2(a^2 + b^2)$ implies

$$\|\nabla H - \nabla H_N^\star\|_{H^{\eta-1}(\Omega)}^2 \leqslant \tilde{C}_H \left( \|H\|_\Psi^2 + \|q\|_{\Gamma^2}^2 + \|p\|_{\Sigma^2}^2 \right). \qquad (15)$$

Putting 13 and 15 into 14, inequality $(a + b)^2 \leqslant 2(a^2 + b^2)$, and using the embeddings $\mathcal{H}_\Sigma^2 \subset \mathcal{H}_\Sigma, \mathcal{H}_\Gamma^2 \subset \mathcal{H}_\Gamma$, results in

$$\|\nabla H - \nabla H_N^\star\|_{L^\infty(\Omega)}^2 \leqslant C_H \left( \varrho_N^{2(\eta - 1 - m)} + \rho_N^{2(\gamma - \gamma')} + \rho_N^{2(\sigma - \sigma')} \right) \left( \|H\|_\Psi^2 + \|q\|_{\Gamma^2}^2 + \|p\|_{\Sigma^2}^2 \right).$$

$$\square$$

**Remark 3.1.** *We can define $\|H\|_{\Psi,0} := \min_{c \in \mathbb{R}} \|H - c\|_\Psi$. Since adding a constant does not affect the gradient, i.e. $\nabla(H - c) = \nabla H$, all of the theoretical steps in the proof remain valid if we replace $\|H\|_\Psi$ by $\|H\|_{\Psi,0}$. In particular, working with $\|H\|_{\Psi,0}$ yields a tighter bound.*

## 4. Implementation

In this section, we expand the methodology introduced in section 2.2. For the 2-step method, we present the closed form solutions of the optimization problems presented in 2.2 and, for the 1-step method, we explain in detail the reduction of the optimization problem and the numerical optimization algorithm implemented for its solution. Then, we discuss kernel and hyperparameter choices and end the discussion with remarks on the energy conservation properties of our methods.

### 4.1. Kernels acting on linear functionals. 
Let $\mathcal{X}$ be any set. Let $\mathcal{H}$ be the RKHS associated to the kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}^\star$ denotes the dual space of $\mathcal{H}$, i.e. it is the space of all bounded linear functionals $\phi : \mathcal{H} \to \mathbb{R}$. For any $x \in \mathcal{X}$, write the kernel section $K_x := K(\cdot, x) \in \mathcal{H}$. By abuse of notation, for a bounded linear functional $\phi \in \mathcal{H}^\star$, we define

$$K(\phi, x) := \phi(K_x) \in \mathbb{R}.$$

Then, naturally, for a vector of functionals $\phi = (\phi_1, \ldots, \phi_N) \in (\mathcal{H}^\star)^N$, we have

$$K(\phi, x) := \phi(K_x) = \big( \phi_1(K_x), \ldots, \phi_N(K_x) \big)^\top \in \mathbb{R}^N.$$

Then, for two vectors of functionals $\phi = (\phi_i)_{i=1}^N$ and $\psi = (\psi_j)_{j=1}^M$, we have

$$K(\phi, \psi) \in \mathbb{R}^{N \times M}, \qquad K(\phi, \psi)_{ij} := \psi_j\big(K(\phi_i, \cdot)\big) = \psi_j\big(K_{\phi_i}\big),$$

where $K(\phi_i, \cdot) \in \mathcal{H}$ satisfies $\langle f, K(\phi_i, \cdot)\rangle_\mathcal{H} = \phi_i(f)$.

**Theorem 4.1** (generalized representer theorem [53, Cor. 17.12])**.** *Let $\mathcal{H}$ be an RKHS associated to the kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and let $\phi = (\phi_1, \ldots, \phi_N) \in (\mathcal{H}^\star)^N$ be a vector of bounded linear functionals on RKHS $\mathcal{H}$. Consider*

$$f^\star := \arg\min_{f \in \mathcal{H}} \|f\|_{\mathcal{H}} \quad s.t. \quad \phi_i(f) = \xi_i, \qquad i = 1, \ldots, N,$$

*for $\xi = (\xi_1, \ldots, \xi_N) \in \mathbb{R}^N$. If $K(\phi, phi)$ is invertible, then every minimizer $f^\star$ has the form*

$$f^\star = K(\phi, \cdot)^\top K(\phi, \phi)^{-1}\xi.$$

*Moreover, it holds that*

$$\|f^\star\|_{\mathcal{H}}^2 + \tfrac{1}{\lambda}\|\phi f - \xi\|_2^2 = \xi^\top (K(\phi, \phi) + \lambda I)^{-1}\xi.$$

**4.2. Common Structure.** We write $\delta_r$ for the bounded point-evaluation functional at $r$; i.e., for a function $f$, $\delta_r(f) = f(r)$. We define two vectors of bounded linear functionals used in derivation of both methods. Further, we assume $\Gamma, \Sigma \in C^1(\mathbb{R} \times \mathbb{R})$ and $\Psi \in C^2(\mathbb{R}^{2m} \times \mathbb{R}^{2m})$ so that the derivative evaluations below are bounded linear functionals on the corresponding RKHSs.

**4.3. Implementing 2-step method.** Denote

$$\phi_{\mathrm{obs}} := (\delta_{s_1}, \ldots, \delta_{s_{N_{\mathrm{obs}}}})^\top \in (\mathcal{H}_\Gamma^\star)^{N_{\mathrm{obs}}} \text{or} (\mathcal{H}_\Sigma^\star)^{N_{\mathrm{obs}}} \text{ (depending on context)}.$$

By the generalized representer theorem (Theorem 4.1) applied with $\mathcal{H} = \mathcal{H}_\Gamma$, $\phi = \phi_{\mathrm{obs}}$, $\xi = q(S)$ and with $\mathcal{H} = \mathcal{H}_\Sigma$, $\phi = \phi_{\mathrm{obs}}$, $\xi = p(S)$ to problems 1 and 2 respectively, their unique interpolants are given by

$$q^\star(\tau) = \Gamma(\tau, S)\, \Gamma(S, S)^{-1}\, q(S), \tag{16}$$

$$p^\star(\tau) = \Sigma(\tau, S)\, \Sigma(S, S)^{-1}\, p(S). \tag{17}$$

Now, given the approximants $q^\star, p^\star$, in order to create input for $H$, since $\Gamma, \Sigma \in C^1$, we can explicitly differentiate these interpolants in time as follows

$$\dot{q}^\star(\tau) = \partial_\tau\Gamma(\tau, S)\, \Gamma(S, S)^{-1}\, q(S), \qquad \dot{p}^\star(\tau) = \partial_\tau\Sigma(\tau, S)\, \Sigma(S, S)^{-1}\, p(S).$$

We will use the stacked derivative vector at collocation times in the form

$$z = \begin{pmatrix} \dot{q}^\star(T) \\ -\dot{p}^\star(T) \end{pmatrix} \in \mathbb{R}^{2Nm},$$

reflecting Hamilton's equations $\dot{q} = \partial_p H$ and $\dot{p} = -\partial_q H$. Denote

$$\varphi := \big(\varphi_{1,1}^{(p)}, \ldots, \varphi_{1,m}^{(p)}, \ldots, \varphi_{N,1}^{(p)}, \ldots, \varphi_{N,m}^{(p)}, \varphi_{1,1}^{(q)}, \ldots, \varphi_{N,m}^{(q)}\big)^\top \in (\mathcal{H}_\Psi^\star)^{2Nm}, \tag{18}$$

where

$$\varphi_{i,j}^{(p)} := \delta_{y(t_i)} \circ \partial_{p_j}, \qquad \varphi_{i,j}^{(q)} := \delta_{y(t_i)} \circ \partial_{q_j}, \qquad i = 1, \ldots, N, \qquad j = 1, \ldots, m.$$

Note that problem 3 can now be rewritten in the equivalent form

$$H^\star = \arg\min_{\hat{H} \in \mathcal{H}_\Psi} \|\hat{H}\|_\Psi^2 + \big\|\Lambda^{-1/2}\big(\varphi(\hat{H}) - z\big)\big\|_2^2, \tag{19}$$

where $\Lambda \in \mathbb{R}^{2Nm \times 2Nm}$ is a block-diagonal matrix defined as

$$\Lambda = \begin{bmatrix} \lambda_1 I_{Nm} & 0 \\ 0 & \lambda_2 I_{Nm} \end{bmatrix}.$$

By the generalized representer theorem (Theorem 4.1) applied with $\mathcal{H} = \mathcal{H}_\Psi$, $\phi = \varphi$, $\xi = z$ to problem 19, its unique minimizer is given by

$$H^\star(x) = \Psi(\varphi, x)^\top \big(\Psi(\varphi, \varphi) + \Lambda\big)^{-1} z, \tag{20}$$

where

$$\Psi\big(\varphi_{i,j}^{(p)}, x\big) \;=\; \partial_{p_j}^y \,\Psi\big(y, x\big)\big|_{y=y(t_i)}, \quad \Psi\big(\varphi_{i,j}^{(q)}, x\big) \;=\; \partial_{q_j}^y \,\Psi\big(y, x\big)\big|_{y=y(t_i)}, \quad i = 1, \ldots, N, \; j = 1, \ldots, m.$$

thus

$$\Psi(\varphi, x) = \begin{pmatrix} \partial_p \Psi\big(y(T), \, x\big) \\ \partial_q \Psi\big(y(T), \, x\big) \end{pmatrix} \in \mathbb{R}^{2Nm},$$

and

$$\Psi(\varphi, \varphi) = \begin{bmatrix} \dfrac{\partial^2 \Psi}{\partial p \, \partial p'}(y(T), y(T)) & \dfrac{\partial^2 \Psi}{\partial p \, \partial q'}(y(T), y(T)) \\[2mm] \dfrac{\partial^2 \Psi}{\partial q \, \partial p'}(y(T), y(T)) & \dfrac{\partial^2 \Psi}{\partial q \, \partial q'}(y(T), y(T)) \end{bmatrix} \in \mathbb{R}^{2Nm \times 2Nm},$$

where each block is an $Nm \times Nm$ matrix.

### 4.3.1. *Computational Bottleneck of the 2-step method.* The two-step approach is the more convenient to implement as it only requires numerically solving linear systems in 16, 17 and 20. Since the matrices $\Gamma(S, S)$ and $\Sigma(S, S)$ are often ill-conditioned, in practice we regularize them by adding a small perturbation defined as a multiple of the identity matrix and invert the perturbed matrices $\Gamma(S, S) + \lambda_q I_{N_{\mathrm{obs}}}$ and $\Sigma(S, S) + \lambda_p I_{N_{\mathrm{obs}}}$ with regularization parameters $\lambda_q, \lambda_p > 0$. The main computational cost is in the inversion of kernel matrices which can be handled using standard numerical solvers for small data sets.

### 4.4. **Implementing 1-step method.** For the 1-step method, since we don't have data on the derivatives $\hat{\dot{q}}(T), \hat{\dot{p}}(T)$, we introduce the latent (slack) time-derivative functions $z_1, z_2 : [0, T_{\mathrm{final}}] \to \mathbb{R}^m$ and rewrite problem 4 as

$$\min_{\hat{q} \in \mathcal{H}_\Gamma, \hat{p} \in \mathcal{H}_\Sigma, \hat{H} \in \mathcal{H}_\Psi, z_1, z_2} \|\hat{H}\|_\Psi^2 + \frac{1}{\lambda_1}\|\hat{q}\|_\Sigma^2 + \frac{1}{\lambda_2}\|\hat{p}\|_\Gamma^2$$

$$\text{subject to} \quad \begin{cases} \hat{q}(S), \hat{\dot{q}}(T) = q(S), z_1(T), \\ \hat{p}(S), \hat{\dot{p}}(T) = p(S), z_2(T), \\ z_1(T), z_2(T) = \partial_p \hat{H}((q, p)(T)), -\partial_q \hat{H}((q, p)(T)). \end{cases}$$

We choose Tikhonov penalties for all constraints and obtain the unconstrained objective

$$\min_{\hat{q} \in \mathcal{H}_\Gamma, \hat{p} \in \mathcal{H}_\Sigma, \hat{H} \in \mathcal{H}_\Psi, z_1, z_2} \|\hat{H}\|_\Psi^2 + \frac{1}{\lambda_1}\|\hat{q}\|_\Sigma^2 + \frac{1}{\lambda_2}\|\hat{p}\|_\Gamma^2 + \frac{1}{\lambda_1}\big\|\Phi\,\hat{q} - (q(S), z_1(T))\big\|_2^2$$

$$+ \frac{1}{\lambda_2}\big\|\Phi\,\hat{p} - (p(S), z_2(T))\big\|_2^2 \tag{21}$$

$$+ \frac{1}{\lambda}\big\|\varphi\hat{H} - (z_1(T), -z_2(T))\big\|_2^2,$$

where $\varphi$ is defined as before in 18 and

$$\Phi := \big(\delta_{s_1}, \ldots, \delta_{s_{N_{\mathrm{obs}}}}, \, \delta_{t_1} \circ \tfrac{\mathrm{d}}{\mathrm{d}t}, \ldots, \delta_{t_N} \circ \tfrac{\mathrm{d}}{\mathrm{d}t}\big)^\top \in \big(\mathcal{H}_\Gamma^\star\big)^{N_{\mathrm{obs}}+N} \text{or} \big(\mathcal{H}_\Sigma^\star\big)^{N_{\mathrm{obs}}+N} \text{ (depending on context)}.$$

For fixed $z_1, z_2$, each term in 21 is a Tikhonov problem in an RKHS with a linear observation operator. By the generalized representer theorem (Theorem 4.1) in its penalized form, minimizers

$q^\star, p^\star, H^\star$ of problem 21 are of the form

$$q^\star(\tau) = \Gamma(\tau, \Phi) \left(\Gamma(\Phi, \Phi) + \lambda_1 I\right)^{-1} \begin{pmatrix} q(S) \\ z_1(T) \end{pmatrix},$$

(22)

$$p^\star(\tau) = \Sigma(\tau, \Phi) \left(\Sigma(\Phi, \Phi) + \lambda_2 I\right)^{-1} \begin{pmatrix} p(S) \\ z_2(T) \end{pmatrix},$$

(23)

$$H^\star(x) = \Psi(x, \varphi) \left(\Psi(\varphi, \varphi) + \lambda I\right)^{-1} \begin{pmatrix} z_1(T) \\ -z_2(T) \end{pmatrix}.$$

(24)

Substituting 22-24 into 21 and using the norm identity in theorem 4.1, we obtain finite-dimensional reduced problem in $z_1, z_2$ as follows

$$\begin{aligned} \min_{z_1, z_2} \quad & \begin{pmatrix} q(S) \\ z_1(T) \end{pmatrix}^\top \left(\Gamma(\Phi, \Phi) + \lambda_1 I\right)^{-1} \begin{pmatrix} q(S) \\ z_1(T) \end{pmatrix} \\ & + \begin{pmatrix} p(S) \\ z_2(T) \end{pmatrix}^\top \left(\Sigma(\Phi, \Phi) + \lambda_2 I\right)^{-1} \begin{pmatrix} p(S) \\ z_2(T) \end{pmatrix} \\ & + \begin{pmatrix} z_1(T) \\ -z_2(T) \end{pmatrix}^\top \left(\Psi(\varphi, \varphi) + \lambda I\right)^{-1} \begin{pmatrix} z_1(T) \\ -z_2(T) \end{pmatrix}. \end{aligned}$$

(25)

Note that $(q^\star, p^\star)(T)$ depend on $(z_1, z_2)$ via 22-23, so 25 is generally nonconvex. Thus, we solve 25 for $(z_1, z_2)$ using a smooth large-scale optimizer (e.g., L-BFGS) and warm-start with the outputs of the 2-step method.

### 4.4.1. *Computational Bottleneck of the 1-step method.* The numerical implementation of the 1-step learning method is significantly more challenging as the objective function couples the estimation of the Hamiltonian $H$ with the dynamics in a single, joint optimization. The main bottleneck here lies in solving the resulting high-dimensional, nonlinear optimization problem over the function space $\mathcal{H}_\Psi$. To address this, we employ a quasi-Newton optimization algorithm, specifically L-BFGS, which is well-suited for large-scale smooth optimization problems. Moreover, to improve convergence and reduce sensitivity to poor initialization, we warm-start the 1-step solver using the output of the 2-step method. This strategy not only accelerates convergence but also improves the stability and reliability of the learned Hamiltonian.

### 4.4.2. *Problem-agnostic CGC Framework* [1]. While it is possible to directly implement the 1-step method described in section 4.4 specifically for Hamiltonian systems, we instead provide a more generic, problem-agnostic framework that implements the general setting of the CGC framework [52]. The framework, written in Python, provides an object-oriented interface for writing the computational graph underlying any CGC problem. It is built on top of JAX [10], which allows the method to run on commercial CPUs or utilize the power of either Nvidia or AMD GPUs to boost the speed and performance of the CGC computation. For more technical explanation, see appendix B.

### 4.5. **Choosing kernels and hyperparameters.** While the 2-step and 1-step learning methods are generic and well-defined for any choice of kernels $\Gamma, \Sigma, \Psi$, the practical performance of these algorithms is closely tied to a good choice of kernels. In general, the choice of these kernels imposes constraints on our model classes for the functions $q, p, H$. Moreover, in many applications we may have access to additional knowledge about the structure of $q, p, H$ and building that knowledge into the kernel will improve the accuracy of our approximation.

**4.5.1.** *Choosing* $\Gamma, \Sigma$ *for* $q, p$. In this work, we assume to have no *a priori* knowledge on the functional form of the trajectories $q, p$. Therefore, in all of our experiments, we take the following Gaussian kernel for approximating $q, p$

$$\Gamma = \Sigma = K, \quad K(t, t') = \exp(-\frac{\|t - t'\|^2}{2\theta^2}),$$

with parameters $\theta = 1.0$, $\lambda = 1e - 5$.

**4.5.2.** *Choosing* $\Psi$ *for* $H$. In a generic situation where there is no prior knowledge beyond smoothness about the structure of the Hamiltonian $H$, it is sensible to take a generic Gaussian kernel for approximating $H$ similarly to our approach for $q, p$. To this end, we considered the Gaussian kernel

$$\Psi(q, p, q', p') = \exp(-\frac{\|q - q'\|^2}{2\theta^2}) \exp(-\frac{\|p - p'\|^2}{2\theta^2}),$$

with parameters $\theta = 1.0$, $\lambda = 0.001$. However, many equations that arise in physical sciences have polynomial forms [20, Sec. 1.2]. In particular, a lot of Hamiltonians are separable polynomials; that is, $H(q, p) = H_1(q) + H_2(p)$ where $H_1, H_2$ are polynomials. In these situations, we can incorporate that information by choosing the associated kernel to be a separable polynomial kernel

$$\Psi(q, p, q', p') = (qq' + \theta)^d + (pp' + \theta)^d,$$

with parameters $\theta = 1.0$, $\lambda = 0.001$. We also chose the polynomials to be cubic, i.e. $d = 3$, as that is usually the maximum degree of polynomial Hamiltonian systems. In cases when $H$ exhibits additive polynomial and non-polynomial structure, we can take a mixed kernel of the form

$$\Psi(q, p, q', p') = \exp(-\frac{\|q - q'\|^2}{2\theta^2}) + (pp' + \theta)^d.$$

## 5. Numerical Experiments

In this section, we numerically illustrate the effectiveness of the proposed methods using a range of benchmark Hamiltonian systems. The purpose of these experiments is to validate both the system identification and forecasting capabilities of the 1-step and 2-step kernel-based methods introduced in Sections 2.2.1 and 2.2.2. In particular, we aim to demonstrate that the 1-step method outperforms the 2-step method in the scarce-data regime, due to its ability to jointly learn the dynamics and the governing Hamiltonian while enforcing the underlying physical structure. Moreover, we numerically compare the efficiency of different kernels for $H$. We consider four representative Hamiltonian systems: the mass-spring system, the two-mass-three-spring system, the Hénon-Heiles system, and the nonlinear pendulum. These systems span a range of complexities, from linear to nonlinear, and from low-dimensional to higher-dimensional and chaotic regimes. For each system, we study the accuracy of trajectory reconstruction, as well as the accuracy of Hamiltonian recovery, and the fidelity of long-term forecasting.

**5.1. Common Structure.** For all the experiments, we follow a common structure for data generation and forecasting and use this common structure for both 2-step and 1-step methods.

**5.1.1.** *Training Data.* Given the explicit form of the true Hamiltonian function $H(y)$, we derive $\frac{dy}{dt}$ and along with the initial condition $y(t_0)$ and the time interval, numerically integrate using the `odeint` scheme from SciPy, which is based on variable-coefficient Adams methods for non-stiff problems, producing our collocation points of $q, p$ evaluated at the given time interval. The integration is performed on a set $T_{col}$ of $N_{col} = 200$ uniformly sampled time points over a fixed time interval from $t = 0$ to $t = 40$ (recall that, in vector form, this is represented as $T$ in the methodology). For sparsity factors $\alpha = 0, 0.5, 0.6, 0.7, 0.8, 0.9$, observed points are randomly subsampled with probability $1 - \alpha$ from the collocation set. The selection of observed points $S_{obs}$ (denoted $S$ in

vector form) is randomized at each experimental run, each experiment is repeated ten times with different random subsamplings, and the results are averaged.

5.1.2. *Testing Data.* For testing, we use the recovered Hamiltonian $H^\star$ and the reconstructed initial state $y^\star(t_0)$ to simulate the system $\frac{dy^\star}{dt}$ on the previously seen interval (interpolation) and forward in time beyond the training window (forecasting). Denote

$$T_{\text{test}} := T_{\text{int}} \cup T_{\text{ext}},$$

to be the *testing set*, where $T_{\text{int}}$ and $T_{\text{ext}}$ are defined as following:

- $T_{\text{int}}$: first part of $T_{\text{test}}$ defined as $T_{\text{int}} := T_{\text{col}} \backslash T_{\text{obs}}$, consisting of the unobserved collocation points ranging from $t = 0$ to $t = 40$, used to test the accuracy of interpolation;
- $T_{\text{ext}}$: second part of $T_{\text{test}}$, consisting of 200 uniformly sampled time points taken from $t = 40$ to $t = 80$, used to test the accuracy of extrapolation (forecasting).

5.1.3. *Evaluation.* Evaluation is performed using both qualitative and quantitative metrics on the testing set. We visualize the learned trajectories over time to compare them with the true dynamics. We also evaluate the relative errors in the recovered trajectories and Hamiltonian, as well as the deviation of the total energy over time. Specifically, the trajectory errors (state errors) are measured using relative error (RE), defined as

$$\text{RE}_q = \frac{\| q - q^\star \|_{L^2(T_{\text{test}})}}{\| q \|_{L^2(T_{\text{test}})}} = \frac{\left( \sum_{t_i \in T_{\text{test}}} \| q(t_i) - q^\star(t_i) \|_2^2 \right)^{\frac{1}{2}}}{\left( \sum_{t_i \in T_{\text{test}}} \| q(t_i) \|_2^2 \right)^{\frac{1}{2}}},$$

for $q$ and similarly for $p$.

5.1.4. *Energy Conservation and Symplectic Integration.* An important characteristic of Hamiltonian systems is their energy conservation. That is, if $H$ has no explicit time dependence, then energy is conserved, i.e. $\frac{dH}{dt} = 0$. Most standard numerical integration schemes usually drift in energy over time. To treat this issue, different variations of symplectic integrators are implemented for Hamiltonians. Here we simply employ `odeint` at a high precision to extrapolate with the learned Hamiltonian, so a symplectic scheme was unnecessary. Therefore, our measure of error in $H$ is our extrapolation error.

5.1.5. *Presentation of Results.* We considered two different regimes through which we compared the two methods numerically: (i) the Gaussian kernel for $H$, and (ii) the separable polynomial kernel for $H$. For all examples, we ran experiments at all sparsity factors mentioned above and reported relative errors for both interpolation and extrapolation. We present the results in the following ways:

- **Trajectory recovery plots:** For demonstration purposes, we only present these plots for sparsity factors 0.5 and 0.8, although results for all values are available in the code repository.
- **Error tables:** These tables report interpolation and extrapolation errors for the 2-step and 1-step methods with Gaussian and polynomial (or additive polynomial Gaussian) kernels, across all sparsity factors. Note that the interpolation errors for the 2-step method (within a single system) are exactly the same regardless of the kernel used for $H$. The reason is that each of $q$ and $p$ is represented with the same Gaussian kernel, independent of the kernel chosen for $H$, as the choice of $H$'s kernel comes into play only in the second step. This is not the case for the 1-step method, where the choice of $H$'s kernel also affects interpolation, since the Hamiltonian constraints are enforced globally on all non-observed points.

- **Error plots:** To quantify these observations, we compute the relative error (RE) for each variable as a function of the sparsity factor. Each plot compares the 2-step and 1-step methods for a fixed kernel across all sparsity factors over the entire time period. This process is performed separately for interpolation and extrapolation. Extrapolation error plots are shown for each example in the main text, while the interpolation error plots are provided in Section C.

## 5.2. **Summary of Results.** Across all systems, kernels, and sparsity levels, several clear patterns emerge.

- **Interpolation vs. Extrapolation:** Interpolation errors are consistently much smaller than extrapolation errors. Both the 2-step and 1-step methods perform well when interpolating within known data regions, where the problem is well-posed; i.e., the approximation error remains bounded within the region covered by the data. In extrapolation settings, as expected from the theory of dynamical forecasting, accuracy decreases since the problem becomes ill-posed and errors amplify under time evolution. Nevertheless, the 1-step method consistently manages this instability better than the 2-step method.
- **2-step vs. 1-step method:** For interpolation, the 1-step method almost always achieves lower errors than the 2-step method. For extrapolation, the 1-step method also tends to outperform the 2-step method, especially at low and moderate sparsity. Even at very high sparsity levels (0.7 and above), where both methods face significant challenges, the 1-step method still yields the smaller errors. This demonstrates the advantage of learning the mapping in a single stage, where the 1-step method avoids error accumulation and simultaneously incorporates both the physics and the data of the problem.
- **Effect of sparsity:** As sparsity increases, errors naturally grow for both interpolation and extrapolation. This trend is expected, since fewer data points reduce the effective constraints on the recovery problem. Importantly, across all sparsity levels, the 1-step method remains more accurate and stable than the 2-step method. The improved robustness comes from the fact that the 1-step formulation directly incorporates the dynamics, whereas the 2-step method loses accuracy through derivative approximations and decoupling of physics from data.
- **Kernel dependence:** With the Gaussian kernel, errors increase faster at high sparsity compared to the separable or additive polynomial kernels. Polynomial-based kernels yield more stable extrapolation errors, and when combined with the 1-step method, provide particularly strong performance. This highlights not only the importance of designing appropriate kernels when prior knowledge is available, but also the ability of the 1-step method to consistently deliver superior results across different kernel choices.

We will now proceed to each individual system.

## 5.3. **Mass-Spring System.** We first consider a harmonic oscillator with Hamiltonian

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

with initial condition $y(t_0) = (0, 1)$. Being linear and one-dimensional, this classical system has periodic motion and serves as a simple test case for validating correctness. Since the Hamiltonian is symmetric with respect to $q$ and $p$, and their recovery behavior is very similar, we moved the recovery plots and the error table for $p$ to the supplementary materials (see Section C).

### 5.3.1. *Trajectory recovery plots.* Figure 3 shows the recovered trajectories for $q$ over time, with sparsity factors 0.5 and 0.8. The ground truth trajectories are plotted in blue, while red crosses denote observed data points. Predictions from the 2-step and 1-step methods are shown with dashed and dash-dotted lines, respectively. At lower sparsity (0.5), both methods provide accurate

approximations. At higher sparsity (0.8), the 1-step method remains stable and tracks the true
dynamics more closely, whereas the 2-step method exhibits noticeable phase drift.
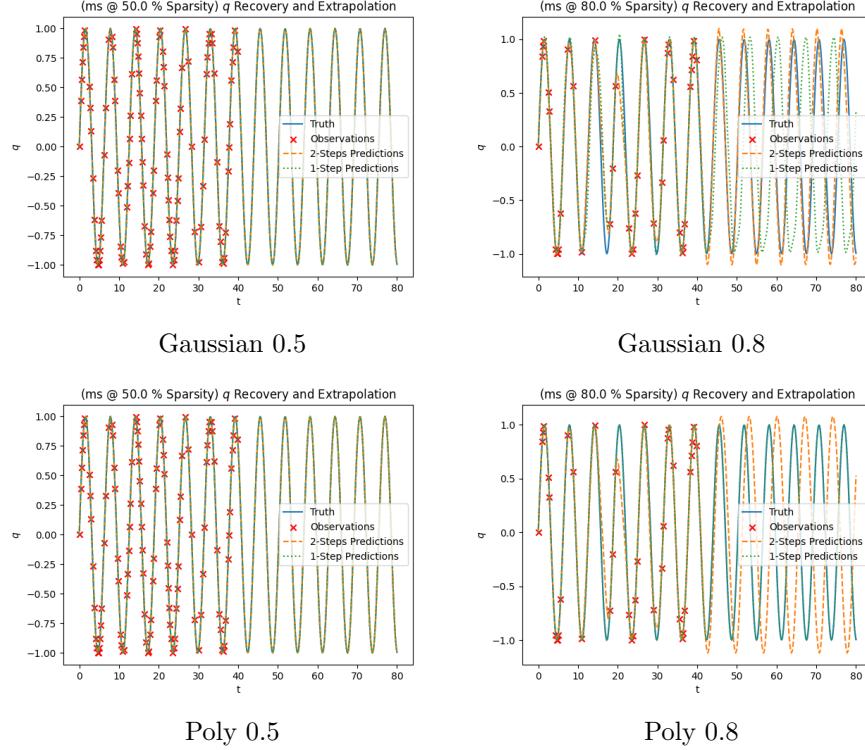


Gaussian 0.5                    Gaussian 0.8

Poly 0.5                        Poly 0.8

FIGURE 3. Recovery of $q$ in the Mass-Spring system.

5.3.2. *Error tables.* Table 1 reports the relative errors for $q$. Two main observations can be drawn
from these results:

(i) 1-step interpolation errors remain close to zero up to moderate sparsity (0.6). Extrapolation
errors grow rapidly with increasing sparsity, but the 1-step method consistently outperforms
the 2-step method. Even at high sparsity levels, where both methods deteriorate, the 1-
step method maintains lower error magnitudes, highlighting the effectiveness of the 1-step
approach under sparse data.

(ii) Within the same method, the separable polynomial kernel yields smaller errors than the
Gaussian kernel, emphasizing the importance of kernel engineering when prior knowledge
is available.

The corresponding table for $p$ is provided in Section C.

5.3.3. *Error plots.* Figure 4 shows the forecasting relative errors for $q$ and $p$, together with shaded
confidence bands representing one standard deviation. For both $q$ and $p$, the 1-step method main-
tains low and stable relative error across all sparsity levels. In contrast, the 2-step method exhibits
rapidly growing errors beyond a sparsity factor of 0.6, indicating its sensitivity to data sparsity.

5.4. **Two-Mass-Three-Spring System.** We now consider a two-dimensional mechanical system
composed of two unit masses connected by three springs. Its nonlinear Hamiltonian is given by

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

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.222 ± 0.000 | 0.005 ± 0.000 | **0.036 ± 0.000** |
| 0.5 | 0.281 ± 0.297 | 1.493 ± 2.218 | **0.025 ± 0.027** | **0.138 ± 0.084** |
| 0.6 | 0.815 ± 0.711 | 8.951 ± 13.545 | **0.195 ± 0.337** | **2.545 ± 5.074** |
| 0.7 | 7.048 ± 7.192 | 59.604 ± 62.958 | **6.282 ± 11.262** | **34.315 ± 66.308** |
| 0.8 | 23.992 ± 14.060 | 104.471 ± 43.668 | **26.322 ± 22.815** | **93.133 ± 54.790** |
| 0.9 | 73.202 ± 19.451 | 141.407 ± 14.438 | **53.002 ± 32.089** | **90.448 ± 56.183** |

Gaussian kernel

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.192 ± 0.000 | 0.005 ± 0.000 | **0.063 ± 0.000** |
| 0.5 | 0.281 ± 0.297 | 1.243 ± 1.106 | **0.018 ± 0.013** | **0.160 ± 0.186** |
| 0.6 | 0.815 ± 0.711 | 3.173 ± 3.762 | **0.037 ± 0.046** | **0.138 ± 0.099** |
| 0.7 | 7.048 ± 7.192 | 17.616 ± 18.914 | **0.051 ± 0.034** | **0.269 ± 0.217** |
| 0.8 | 23.992 ± 14.060 | 79.531 ± 42.974 | **0.108 ± 0.070** | **0.286 ± 0.243** |
| 0.9 | 73.202 ± 19.451 | 147.251 ± 8.506 | **0.077 ± 0.026** | **0.170 ± 0.087** |

Separable polynomial kernel

TABLE 1. Relative errors (mean ± std) for $q$ in the Mass-Spring system.

The initial condition is fixed at $y(t_0) = (0.2, -0.1, 0.1, -0.1)$. The system exhibits coupled oscillatory behavior and is a standard multi-variable benchmark.

5.4.1. *Trajectory recovery plots.* Similar to the previous system, in Figure 5 we observe good recovery at sparsity 0.5 from both methods, but in the 0.8 sparsity regime, the 1-step method outperforms the 2-step method, especially in the forecasting region. Since the Hamiltonian is symmetric with respect to $q_1$ and $q_2$, and it is an order-two polynomial in $q_1, q_2, p_1, p_2$, we only report the trajectory plots for $q_1$ here and provide the rest in Section C.

5.4.2. *Error tables.* As in the previous example (Table 2), especially in the forecasting regime, increasing sparsity shows that the 1-step method outperforms the 2-step method. The only caveat is that due to numerical instabilities at sparsity 0.9 with the polynomial kernel (specifically matrix inversion issues), the tables only report up to sparsity 0.8 for this kernel.

5.4.3. *Error plots.* The forecasting error plots in Figure 6 show a trend similar to the previous system: starting from sparsity 0.6, the 2-step method's error grows significantly compared to the 1-step method.

5.5. **Hénon-Heiles System.** As our last example, we consider the Hénon-Heiles system, a well-known two-dimensional nonlinear chaotic Hamiltonian system. Its Hamiltonian is given by

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

We fix the initial condition as $y(t_0) = (0.2, -0.1, 0.1, -0.1)$. The 1-step method is able to recover the dynamics and forecast future behavior with reasonable accuracy, even in the presence of chaotic features. Naturally, the prediction horizon is limited due to the system's sensitivity to initial conditions.
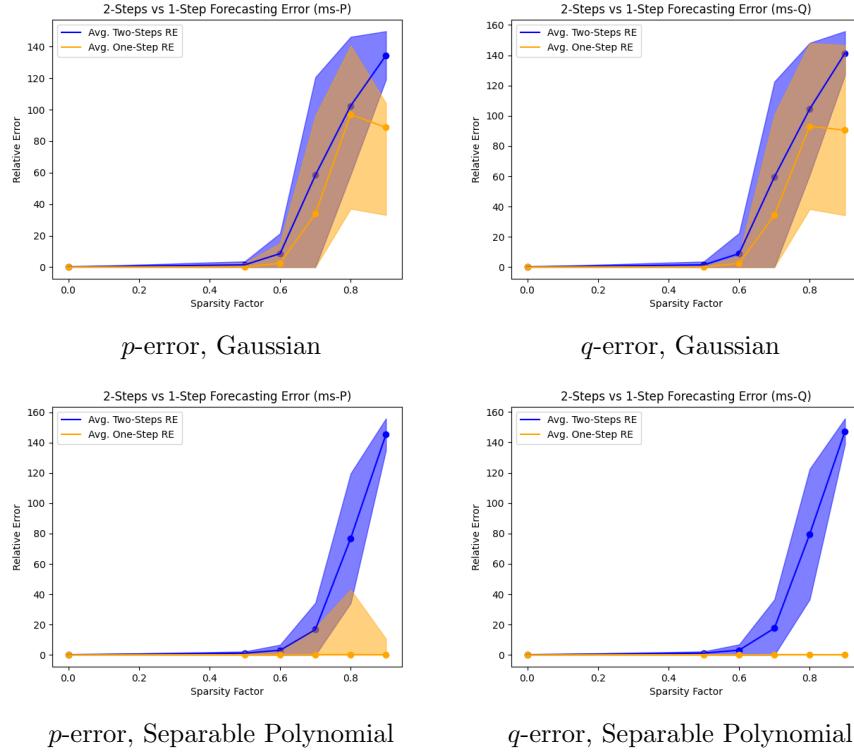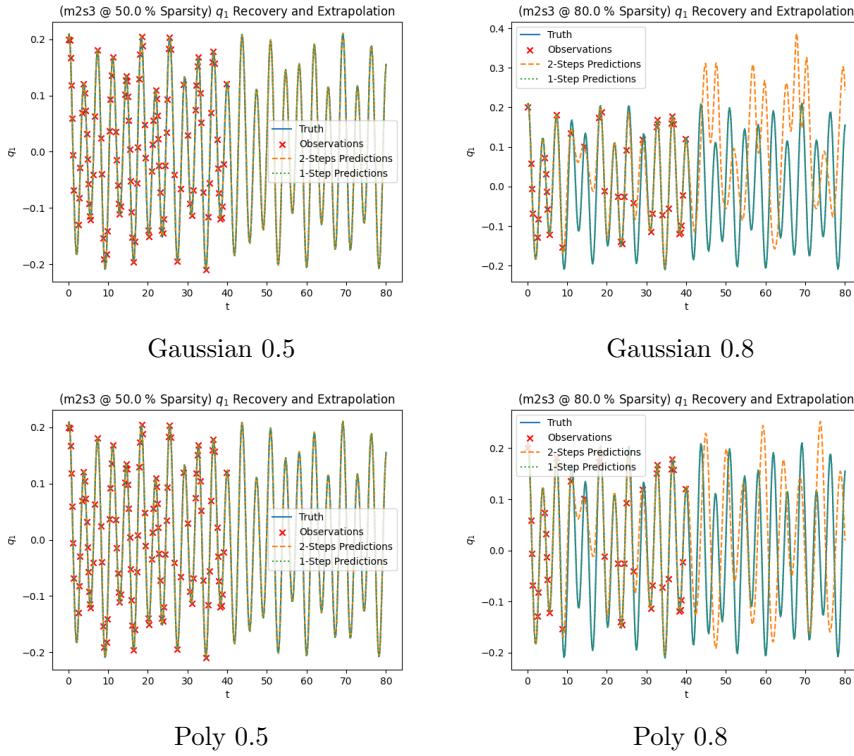
$p$-error, Gaussian

$q$-error, Gaussian

$p$-error, Separable Polynomial

$q$-error, Separable Polynomial

FIGURE 4. Forecasting relative error for the Mass-Spring System.



Gaussian 0.5

Gaussian 0.8

Poly 0.5

Poly 0.8

FIGURE 5. Recovery of $q_1$ in the Two-Mass-Three-Spring system.

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.203 ± 0.000 | 0.168 ± 0.000 | **0.224 ± 0.000** |
| 0.5 | 0.360 ± 0.285 | 1.451 ± 1.290 | **0.072 ± 0.069** | **0.214 ± 0.082** |
| 0.6 | 0.920 ± 0.796 | 5.396 ± 7.568 | **0.037 ± 0.010** | **0.194 ± 0.081** |
| 0.7 | 15.480 ± 12.938 | 54.705 ± 57.873 | **1.287 ± 2.688** | **5.146 ± 11.018** |
| 0.8 | 42.990 ± 20.771 | 122.531 ± 43.043 | **7.603 ± 9.835** | **17.316 ± 23.083** |
| 0.9 | 103.826 ± 22.026 | 130.073 ± 18.070 | **58.210 ± 32.230** | **79.855 ± 25.081** |

Gaussian kernel

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.199 ± 0.000 | 0.011 ± 0.000 | **0.157 ± 0.000** |
| 0.5 | 0.360 ± 0.285 | 1.545 ± 1.548 | **0.030 ± 0.016** | **0.218 ± 0.126** |
| 0.6 | 0.920 ± 0.796 | 5.811 ± 8.604 | **0.061 ± 0.052** | **0.300 ± 0.211** |
| 0.7 | 15.480 ± 12.938 | 71.540 ± 57.892 | **0.104 ± 0.053** | **0.356 ± 0.181** |
| 0.8 | 42.990 ± 20.771 | 134.576 ± 35.819 | **0.156 ± 0.072** | **0.401 ± 0.231** |

Separable polynomial kernel

TABLE 2. Relative errors (mean ± std) for $q$ in the Two-Mass-Three-Spring system.

5.5.1. *Trajectory recovery plots.* Figures 7 and 8 display the recovered trajectories for $q_1$ and $q_2$, respectively, under sparsity factors 0.5 and 0.8. We showed these two here as $q_2$ here is of order 3 and reported the plots for the rest of variables in C. At moderate sparsity (0.5), both methods perform reasonably well. At higher sparsity (0.8), the 1-step method continues to follow the true dynamics more closely, while the 2-step method deviates significantly, particularly in the forecasting region.

5.5.2. *Error tables.* Table 3 reports the relative errors for $q$. Interpolation with the 1-step method is significantly more accurate than with the 2-step method, especially for sparsities 0.5–0.7. Extrapolation shows the same trend: the 1-step method consistently yields smaller errors, though both methods exhibit large errors at high sparsity. Polynomial kernels mitigate extrapolation blow-up more effectively than Gaussian kernels.

5.5.3. *Error plots.* Figure 9 shows the forecasting relative errors for both $q$ and $p$. Once again, the 1-step method maintains lower errors across sparsity levels, while the 2-step method degrades rapidly beyond sparsity 0.6. The separable polynomial kernel provides more stable performance in the extrapolation regime compared to the Gaussian kernel.

5.6. **Nonlinear Pendulum.** For our next example, we consider a one-dimensional nonlinear system governed by the Hamiltonian

$$H(q, p) = \frac{1}{2}p^2 - \cos(q),$$

with initial condition $y(t_0) = (0.95\pi, 0.0)$. This system exhibits nonlinear oscillations and highlights the capability of our method to handle non-polynomial Hamiltonians.

5.6.1. *Trajectory recovery plots.* Unlike the previous examples, here the recovery behavior differs between $q$ and $p$. Therefore we report both variables in the main body (Figures 11 and 10). Despite the difference, once again, at moderate sparsity (0.5), both the 1-step and 2-step methods provide
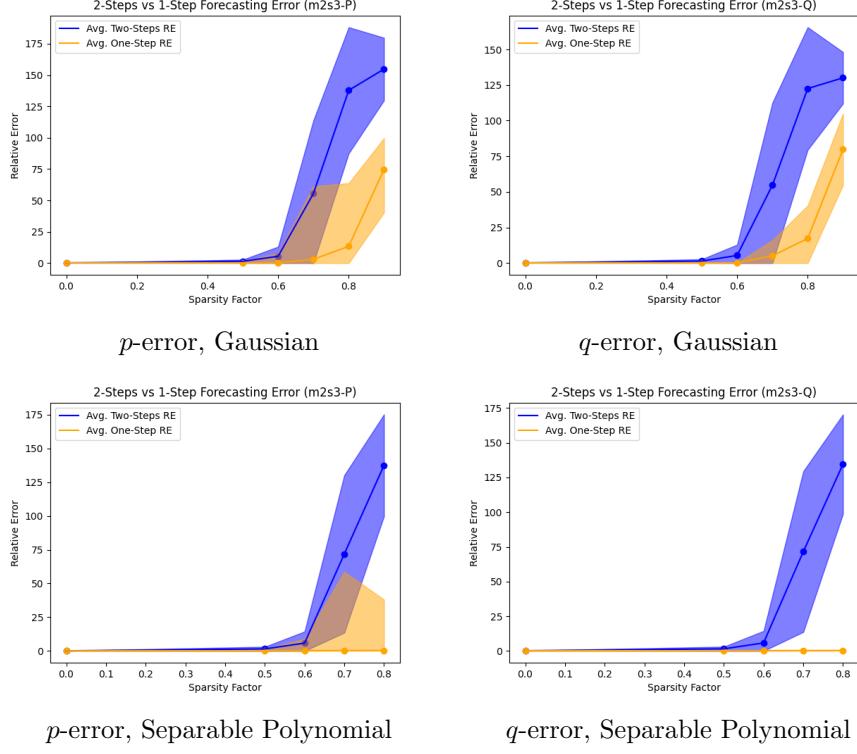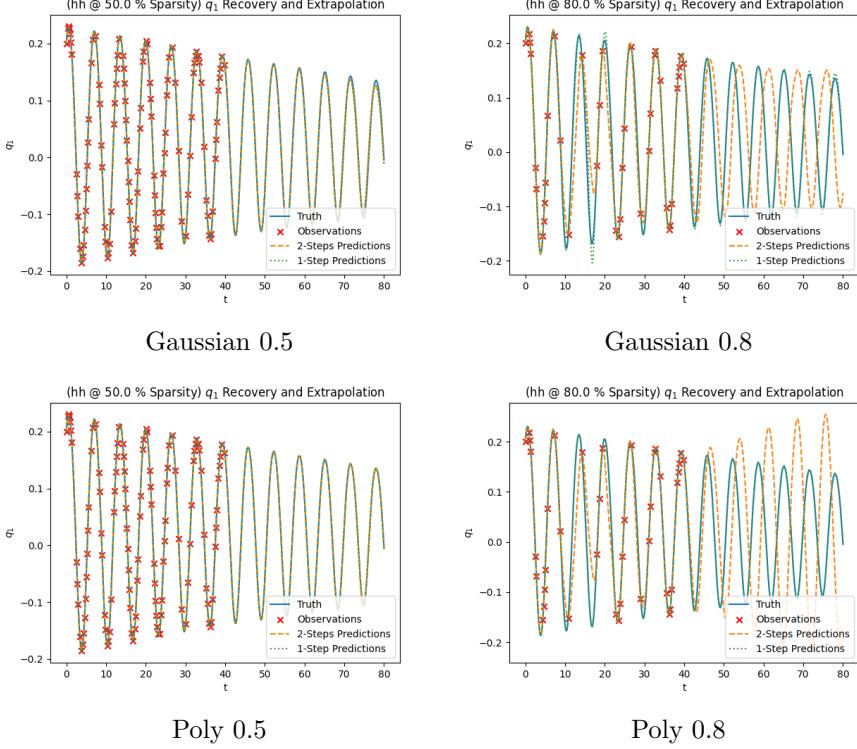
$p$-error, Gaussian          $q$-error, Gaussian

$p$-error, Separable Polynomial          $q$-error, Separable Polynomial

FIGURE 6. Forecasting relative errors for the Two-Mass-Three-Spring system.



Gaussian 0.5          Gaussian 0.8

Poly 0.5          Poly 0.8

FIGURE 7. Recovery of $q_1$ in the Hénon-Heiles system.

Gaussian 0.5

Gaussian 0.8

Poly 0.5

Poly 0.8

FIGURE 8. Recovery of $q_2$ in the Hénon-Heiles system.



$p$-error, Gaussian

$q$-error, Gaussian

$p$-error, Separable Polynomial

$q$-error, Separable Polynomial

FIGURE 9. Forecasting relative errors for the Hénon-Heiles System.

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 2.977 ± 0.000 | 0.002 ± 0.000 | **2.988 ± 0.000** |
| 0.5 | 0.103 ± 0.097 | 2.842 ± 0.371 | **0.022 ± 0.015** | 3.139 ± 0.158 |
| 0.6 | 0.303 ± 0.197 | 3.600 ± 1.511 | **0.040 ± 0.022** | **3.141 ± 0.333** |
| 0.7 | 5.123 ± 4.006 | 17.517 ± 14.945 | **1.486 ± 3.676** | **8.332 ± 10.159** |
| 0.8 | 22.769 ± 14.063 | 64.279 ± 40.605 | **7.272 ± 8.807** | **21.508 ± 27.138** |
| 0.9 | 69.056 ± 20.458 | 135.651 ± 13.299 | **28.934 ± 20.882** | **46.749 ± 29.029** |

Gaussian kernel

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 1.145 ± 0.000 | 0.002 ± 0.000 | **1.145 ± 0.000** |
| 0.5 | 0.103 ± 0.097 | 1.790 ± 0.747 | **0.013 ± 0.007** | **1.252 ± 0.108** |
| 0.6 | 0.303 ± 0.197 | 3.190 ± 2.567 | **0.028 ± 0.024** | **1.197 ± 0.057** |
| 0.7 | 5.123 ± 4.006 | 17.339 ± 17.574 | **0.036 ± 0.022** | **1.360 ± 0.199** |
| 0.8 | 22.769 ± 14.063 | 74.536 ± 43.707 | **0.059 ± 0.031** | **1.434 ± 0.217** |
| 0.9 | 69.056 ± 20.458 | 144.810 ± 12.041 | **0.117 ± 0.095** | **1.856 ± 0.528** |

Separable polynomial kernel

TABLE 3. Relative errors (mean ± std) for $q$ in the Hénon-Heiles system.

good recovery for $q$ and $p$; at higher sparsity (0.8), however, the 1-step method remains stable and continues to track the true dynamics more accurately, while the 2-step method shows larger deviations, particularly in the forecasting region.

5.6.2. *Error tables.* Tables 4 and 5 report the relative errors for $q$ and $p$, respectively. In both interpolation and extrapolation, the 1-step method consistently outperforms the 2-step method. For the Gaussian kernel, extrapolation errors increase sharply at high sparsity. In contrast, the additive polynomial + Gaussian kernel provides more stable performance in the extrapolation regime.

5.6.3. *Error plots.* Figure 12 shows the forecasting relative errors for $q$ and $p$. Here again, the 1-step method maintains low errors across all sparsity levels, whereas the 2-step method shows rapidly increasing errors beyond sparsity 0.6. This further highlights the robustness of the 1-step approach under sparse data conditions.

## 6. **Concluding remarks**

In this work, we developed kernel-based methodologies, inspired by the CGC and KEqL frameworks [52, 36], for learning Hamiltonian dynamical systems from time-series data, addressing both system identification and forecasting tasks. We introduced a principled approach to approximate unknown trajectories and Hamiltonians using RKHS theory. Our results show that the simultaneous learning method consistently outperforms the standard kernel based akin to our two-step approach, particularly in data-scarce regimes, while automatically preserving the geometric structure of Hamiltonian systems. We also established theoretical error rates that ensure the reliability of the learned models. In addition, the implementation we provide is general and can be adapted to other CGC [52] problems beyond Hamiltonian dynamics. Taken together, the interpretability,
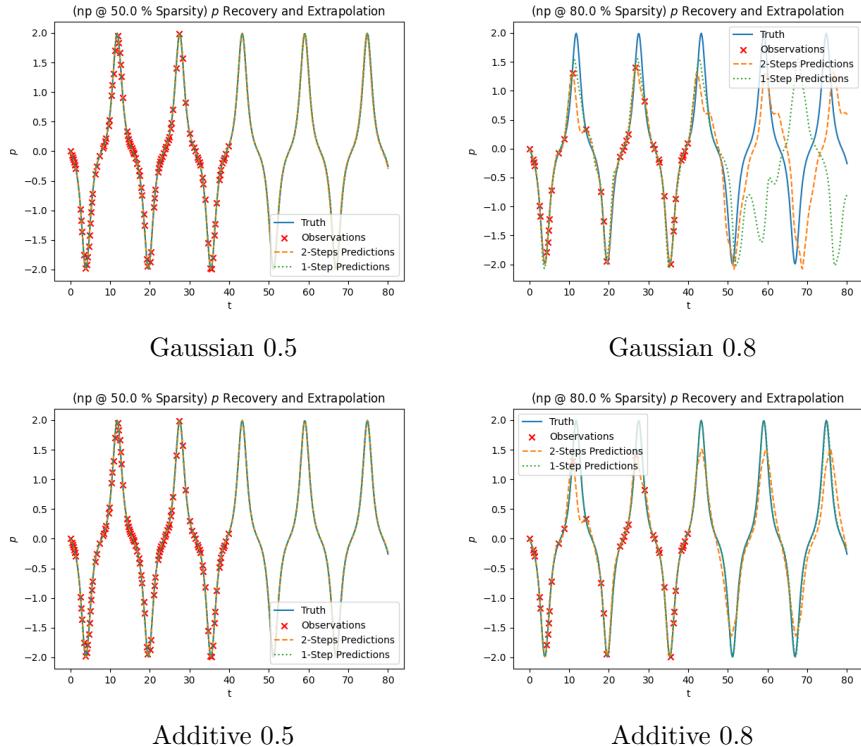
Gaussian 0.5

Gaussian 0.8

Additive 0.5

Additive 0.8

FIGURE 10. Recovery of $p$ in the Nonlinear Pendulum system.



Gaussian 0.5

Gaussian 0.8

Additive 0.5

Additive 0.8

FIGURE 11. Recovery of $q$ in the Nonlinear Pendulum system.

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.234 ± 0.000 | 0.008 ± 0.000 | **1.872 ± 0.000** |
| 0.5 | 0.182 ± 0.148 | 2.549 ± 3.069 | **0.035 ± 0.046** | **1.136 ± 0.980** |
| 0.6 | 0.599 ± 0.575 | 5.061 ± 5.205 | **0.085 ± 0.120** | **2.062 ± 1.948** |
| 0.7 | 5.481 ± 4.974 | 45.986 ± 48.646 | **1.013 ± 1.528** | **14.267 ± 38.004** |
| 0.8 | 15.302 ± 10.281 | 102.954 ± 47.515 | **9.592 ± 11.503** | **74.506 ± 67.847** |
| 0.9 | 59.719 ± 16.933 | 127.130 ± 22.565 | **33.419 ± 16.313** | **97.625 ± 54.474** |

Gaussian kernel

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.228 ± 0.000 | 0.011 ± 0.000 | **1.890 ± 0.000** |
| 0.5 | 0.182 ± 0.148 | 8.016 ± 7.455 | **0.013 ± 0.003** | **1.046 ± 0.529** |
| 0.6 | 0.599 ± 0.575 | 32.859 ± 20.962 | **0.014 ± 0.004** | **1.337 ± 0.827** |
| 0.7 | 5.481 ± 4.974 | 81.800 ± 46.374 | **0.025 ± 0.024** | **2.192 ± 2.705** |
| 0.8 | 15.302 ± 10.281 | 80.487 ± 41.416 | **0.051 ± 0.069** | **3.056 ± 2.150** |
| 0.9 | 59.719 ± 16.933 | 122.773 ± 29.940 | **25.260 ± 15.747** | **102.912 ± 53.001** |

Additive polynomial + Gaussian kernel

TABLE 4. Relative errors (mean ± std) for $q$ in the Nonlinear Pendulum system.

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.336 ± 0.000 | 0.014 ± 0.000 | **2.655 ± 0.000** |
| 0.5 | 0.267 ± 0.188 | 3.583 ± 4.310 | **0.101 ± 0.091** | **1.610 ± 1.390** |
| 0.6 | 0.824 ± 0.627 | 7.179 ± 7.248 | **0.144 ± 0.120** | **2.922 ± 2.757** |
| 0.7 | 6.008 ± 5.373 | 48.595 ± 46.310 | **1.412 ± 2.069** | **14.468 ± 36.524** |
| 0.8 | 23.172 ± 22.818 | 100.743 ± 43.798 | **17.445 ± 17.640** | **71.424 ± 60.946** |
| 0.9 | 65.326 ± 36.965 | 239.710 ± 329.837 | **46.574 ± 18.174** | **105.368 ± 40.820** |

Gaussian kernel

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.324 ± 0.000 | 0.024 ± 0.000 | **2.681 ± 0.000** |
| 0.5 | 0.267 ± 0.188 | 11.231 ± 10.373 | **0.049 ± 0.017** | **1.484 ± 0.752** |
| 0.6 | 0.824 ± 0.627 | 42.634 ± 25.174 | **0.065 ± 0.020** | **1.898 ± 1.174** |
| 0.7 | 6.008 ± 5.373 | 85.578 ± 41.625 | **0.093 ± 0.046** | **3.108 ± 3.831** |
| 0.8 | 23.172 ± 22.818 | 88.899 ± 39.267 | **0.182 ± 0.171** | **4.336 ± 3.048** |
| 0.9 | 65.326 ± 36.965 | 140.867 ± 32.271 | **37.796 ± 23.027** | **102.901 ± 43.392** |

Additive polynomial + Gaussian kernel

TABLE 5. Relative errors (mean ± std) for $p$ in the Nonlinear Pendulum system.

theoretical guarantees, and structure-preserving properties of our methods make them a compelling alternative to existing machine learning approaches.

*p*-error, Gaussian       *q*-error, Gaussian

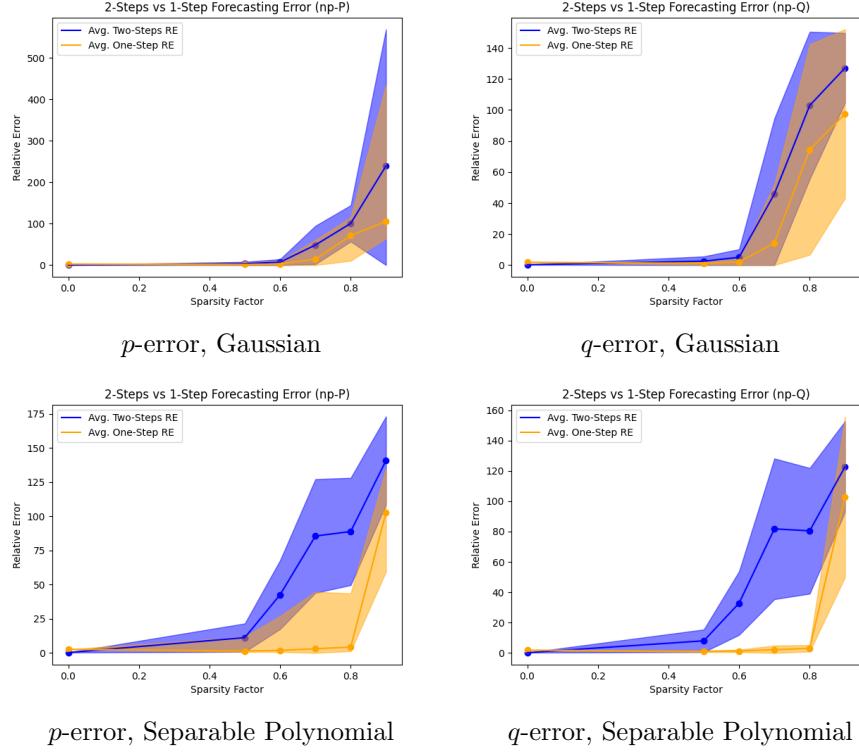*p*-error, Separable Polynomial    *q*-error, Separable Polynomial

FIGURE 12. Forecasting relative error for the Nonlinear Pendulum System.

Future directions include extending this framework to larger-scale and higher-dimensional Hamiltonian systems, and developing adaptive kernel designs that automatically adjust to the specific characteristics of the underlying system to improve data efficiency and scalability.

## 7. **Acknowledgment**

## References

[1] H. Abarbanel. *Analysis of Observed Chaotic Data*. Institute for Nonlinear Science. Springer New York, 2012.

[2] R. A. Adams and J. J. F. Fournier. *Sobolev Spaces*, volume 140. Academic press, 2003.

[3] Romeo Alexander and Dimitrios Giannakis. Operator-theoretic framework for forecasting nonlinear time series with kernel analog techniques. *Physica D: Nonlinear Phenomena*, 409:132520, 2020.

[4] Pau Batlle, Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Error analysis of kernel/gp methods for nonlinear and parametric pdes, 2023.

[5] Tom Bertalan, Felix Dietrich, Igor Mezić, and Ioannis G. Kevrekidis. On learning hamiltonian systems from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12):121107, 2019.

[6] Jake Bouvrie and Boumediene Hamzi. Balanced reduction of nonlinear control systems in reproducing kernel hilbert space. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 294–301, 2010.

[7] Jake Bouvrie and Boumediene Hamzi. Empirical estimators for stochastically forced nonlinear systems: Observability, controllability and the invariant measure. *Proc. of the 2012 American Control Conference*, pages 294–301, 2012. https://arxiv.org/abs/1204.0563v1.

[8] Jake Bouvrie and Boumediene Hamzi. Kernel methods for the approximation of nonlinear systems. *SIAM J. Control and Optimization*, 2017. https://arxiv.org/abs/1108.2903.

[9] Jake Bouvrie and Boumediene Hamzi. Kernel methods for the approximation of some key quantities of nonlinear systems. *Journal of Computational Dynamics*, 1, 2017. http://arxiv.org/abs/1204.0563.

[10] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[11] Steven L. Brunton, Joshua L. Proctor, and J. Nathan 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.

[12] Jan Brüdigam, Martin Schuck, Alexandre Capone, Stefan Sosnowski, and Sandra Hirche. Structure-preserving learning using gaussian processes and variational integrators, 2022.

[13] Martin Casdagli. Nonlinear prediction of chaotic time series. *Physica D: Nonlinear Phenomena*, 35(3):335 – 356, 1989.

[14] Elena Celledoni, Andrea Leone, Davide Murari, and Brynjulf Owren. Learning hamiltonians of constrained mechanical systems, 2022.

[15] Ashesh Chattopadhyay, Pedram Hassanzadeh, Krishna V. Palem, and Devika Subramanian. Data-driven prediction of a multi-scale lorenz 96 chaotic system using a hierarchy of deep learning methods: Reservoir computing, ann, and RNN-LSTM. *CoRR*, abs/1906.08829, 2019.

[16] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Solving and learning nonlinear pdes with gaussian processes. *arXiv preprint arXiv:2103.12959*, 2021.

[17] Yuhan Chen, Takashi Matsubara, and Takaharu Yaguchi. Neural symplectic form: Learning hamiltonian equations on general coordinate systems. In *Advances in Neural Information Processing Systems*, 2021.

[18] Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. In *International Conference on Learning Representations*, 2020.

[19] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2002.

[20] Lawrence C. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2022.

[21] Edward Fuselier and Grady B Wright. Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates. *SIAM Journal on Numerical Analysis*, 50(3):1753–1776, 2012.

[22] Peter Giesl, Boumediene Hamzi, Martin Rasmussen, and Kevin Webster. Approximation of Lyapunov functions from noisy data. *Journal of Computational Dynamics*, 7:57–81, 2019. https://arxiv.org/abs/1601.01568.

[23] R. González-García, R. Rico-Martínez, and I.G. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & Chemical Engineering*, 22:S965–S968, 1998. European Symposium on Computer Aided Process Engineering-8.

[24] Ove Grandstrand. Nonlinear system identification using neural networks: dynamics and instanbilities. In A. B. Bulsari, editor, *Neural Networks for Chemical Engineers*, chapter 16, pages 409–442. Elsevier, Elsevier, 1995.

[25] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, 2019.

[26] Anthony Gruber, Kookjin Lee, and Nathaniel Trask. Reversible and irreversible bracket-based dynamics for deep graph neural networks, 2023.

[27] Bernard Haasdonk, Boumediene Hamzi, Gabriele Santin, and Dominik Wittwar. *Greedy Kernel Methods for Center Manifold Approximation*, page 95–106. Springer International Publishing, 2020.

[28] Bernard Haasdonk, Boumediene Hamzi, Gabriele Santin, and Dominik Wittwar. Kernel methods for center manifold approximation and a weak data-based version of the center manifold theorems. *Physica D*, 2021.

[29] Boumediene Hamzi and Fritz Colonius. Kernel methods for the approximation of discrete-time linear autonomous and control systems. *SN Applied Sciences*, 1(7):1–12, 2019.

[30] Boumediene Hamzi, Amirhossein Jafarian, Houman Owhadi, and Léo Paillet. A Note on Microlocal Kernel Design for Some Slow-Fast Stochastic Differential Equations with Critical Transitions and Application to EEG Signals, 2022.

[31] Boumediene Hamzi, Christian Kuehn, and Sameh Mohamed. A note on kernel methods for multiscale systems with critical transitions. *Mathematical Methods in the Applied Sciences*, 42(3):907–917, 2019.

[32] Boumediene Hamzi, Umesh G. Vaidya, and Houman Owhadi. Kernel methods for some transport equations with application to learning kernels for the approximation of koopman eigenfunctions: A unified approach via variational methods, green's functions, and rkhs. https://www.researchgate.net/publication/391449679, May 2025. Preprint.

[33] Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ode models with gaussian processes. In *International Conference on Machine Learning*, pages 1959–1968. PMLR, 2018.

[34] Boya Hou, Amarsagar Reddy Ramapuram Matavalam, Subhonmesh Bose, and Umesh Vaidya. Propagating uncertainty through system dynamics in reproducing kernel hilbert space. *Physica D: Nonlinear Phenomena*, page 134168, 2024.

[35] John L. Hudson, M. Kube, Raymond A. Adomaitis, Ioannis G. Kevrekidis, Alan S. Lapedes, and R.M. Farber. Nonlinear signal processing and system identification: Applications to time series from electrochemical reactions. *Chemical Engineering Science*, 45(8):2075–2081, 1990.

[36] Yasamin Jalalian, Juan Felipe Osorio Ramirez, Alexander Hsu, Bamdad Hosseini, and Houman Owhadi. Data-efficient kernel methods for learning differential equations and their solution operators: Algorithms and error analysis, 2025.

[37] Pengzhan Jin, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. Sympnets: Intrinsic structure-preserving symplectic networks for identifying hamiltonian systems, 2020.

[38] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences, 2018.

[39] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, USA, 1997.

[40] Victoria Klein et al. Data-driven discovery of coordinates and governing equations. *arXiv preprint arXiv:2203.12610*, 2022.

[41] Stefan Klus, Feliks Nuske, and Boumediene Hamzi. Kernel-based approximation of the Koopman generator and Schrödinger operator. *Entropy*, 22, 2020. https://www.mdpi.com/1099-4300/22/7/722.

[42] Stefan Klus, Feliks Nüske, Sebastian Peitz, Jan-Hendrik Niemann, Cecilia Clementi, and Christof Schütte. Data-driven approximation of the koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020.

[43] Andreas Bittracher, Stefan Klus, Boumediene Hamzi, Peter Koltai, and Christof Schutte. Dimensionality reduction of complex metastable systems via kernel embeddings of transition manifold, 2019. https://arxiv.org/abs/1904.08622.

[44] Jonghyeon Lee, Boumediene Hamzi, Boya Hou, Houman Owhadi, Gabriele Santin, and Umesh Vaidya. Kernel methods for the approximation of the eigenfunctions of the koopman operator. *Physica D: Nonlinear Phenomena*, 476:134662, 2025. Available online 17 April 2025, Version of Record 18 April 2025.

[45] Jonghyeon Lee, Boumediene Hamzi, Yannis Kevrekidis, and Houman Owhadi. Gaussian processes simplify differential equations, September 2024.

[46] Jonghyeon Lee, Houman Owhadi, Boumediene Hamzi, and Umesh G. Vaidya. A note on kernel methods for the construction of lyapunov functions using koopman eigenfunctions. https://www.researchgate.net/publication/387344850, December 2024. Preprint.

[47] Daniel Lengyel, Boumediene Hamzi, Houman Owhadi, and Panos Parpas. Kernel sum of squares for data adapted kernel learning of dynamical systems from data: A global optimization approach. *arXiv preprint arXiv:2408.06465*, 2024.

[48] Ziming Liu, Varun Madhavan, and Max Tegmark. Machine learning conservation laws from differential equations. *Physical Review E*, 106(4), oct 2022.

[49] Da Long, Nicole Mrvaljevic, Shandian Zhe, and Bamdad Hosseini. A kernel approach for pde discovery and operator learning, 2023.

[50] Francis Narcowich, Joseph Ward, and Holger Wendland. Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting. *Mathematics of Computation*, 74(250):743–763, 2005.

[51] A. Nielsen. *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*. O'Reilly Media, 2019.

[52] Houman Owhadi. Computational Graph Completion. *Research in the Mathematical Sciences*, 9(2)(27), 2021.

[53] Houman Owhadi and Clint Scovel. *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*, volume 35. Cambridge University Press, 2019.

[54] Jaideep Pathak, Zhixin Lu, Brian R. Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[55] Ramiro Rico-Martinez, K Krischer, IG Kevrekidis, MC Kube, and JL Hudson. Discrete-vs. continuous-time nonlinear signal processing of cu electrodissolution data. *Chemical Engineering Communications*, 118(1):25–48, 1992.

[56] Gabriele Santin and Bernard Haasdonk. Kernel methods for surrogate modeling. *System and Data-Driven Methods and Algorithms*, 2019. https://arxiv.org/abs/1907.105566.

[57] Harsh Sharma, Juan Diego Draxl Giannoni, and Boris Kramer. Structure-preserving lift & learn: Scientific machine learning for nonlinear conservative partial differential equations, 2025.

[58] Alexandre Smirnov, Boumediene Hamzi, and Houman Owhadi. Mean-field limits of trained weights in deep learning: A dynamical systems perspective. *Dolomites Research Notes on Approximation*, 15(3), 2022.

[59] Riccardo Valperga et al. Structure-preserving learning using gaussian processes and variational integrators. *arXiv preprint arXiv:2112.05451*, 2021.

[60] Riccardo Valperga, Kevin Webster, Victoria Klein, Dmitry Turaev, and Jeroen S. W. Lamb. Learning reversible symplectic dynamics, 2022.

[61] Riccardo Valperga, Kevin Webster, Dmitry Turaev, Victoria Klein, and Jeroen Lamb. Learning reversible symplectic dynamics. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pages 906–916. PMLR, 2022.

[62] Alan John Varghese, Zhen Zhang, and George Em Karniadakis. Sympgnns: Symplectic graph neural networks for identifying high-dimensional hamiltonian systems and node classification, 2024.

[63] Holger Wendland. *Scattered Data Approximation*. Cambridge University Press, 2004.

[64] Holger Wendland and Christian Rieger. Approximate interpolation with applications to selecting smoothing parameters. *Numerische Mathematik*, 101:729–748, 2005.

## Appendix A. Background Results for the Theory

In this section, we provide results used in the proof of theorem 3.1, which also justify the need for some of the assumptions needed for the theorem to hold in section 3. We assume the reader is familiar with classical results in kernel methods and RKHS theory, as well as classical results of functional analysis. The results in this appendix are classical in approximation theory with kernel methods, but the formulations are adapted from [36].

### A.1. Nested RKHS.

**Proposition A.1.** *Suppose $\Omega \subset \mathbb{R}^q$ is bounded and let $K$ be a positive-definite kernel that is continuous in both of its arguments on $\overline{\Omega}$. Then there exists an orthonormal set of continuous eigenfunctions $\{e_i\}_{i=1}^\infty \subset L^2(\Omega)$ and decreasing eigenvalues $\{\lambda_i\}_{i=1}^\infty$, $\lambda_1 \geqslant \lambda_2 \geqslant \ldots$, such that $K(x,x') = \sum_{i=1}^\infty \lambda_i e_i(x) e_i(x')$, and the RKHS $\mathcal{H}$ can be characterized as*

$$\mathcal{H} = \left\{ f : \Omega \to \mathbb{R} \middle| f(x) = \sum_{i \in \{i | \lambda_i \neq 0\}} c_i(f) e_i(x), \quad \sum_{i \in \{i | \lambda_i \neq 0\}} \lambda_i^{-1} c_i(f)^2 < +\infty \right\} \quad (26)$$

*and for any pair $f, f' \in \mathcal{H}$ we have $\langle f, f' \rangle_\mathcal{H} = \sum_{i \in \{i | \lambda_i \neq 0\}} \lambda_i^{-1} c_i(f) c_i(f')$.*

**Definition A.1.** *Given 26, we further define the nested ladder of RKHS spaces*

$$\mathcal{H}^\gamma := \left\{ f : \Omega \to \mathbb{R} \middle| f(x) = \sum_{i=1}^\infty c_i(f) e_i(x), \quad \sum_{i=1}^\infty \lambda_i^{-\gamma} c_i(f)^2 < +\infty \right\},$$

*for $\gamma \geqslant 1$. Naturally, larger values of $\gamma$ imply more "smoothness", in particular we have the inclusion $\mathcal{H}^{\gamma_2} \subset \mathcal{H}^{\gamma_1}$ for $1 \leqslant \gamma_1 < \gamma_2$.*

**Lemma A.1.** *Suppose $f \in \mathcal{H}^{2\gamma}$ and $f' \in \mathcal{H}^\gamma$. Then*

$$\langle f', f \rangle_{\mathcal{H}^\gamma} \leqslant \|f'\|_{L^2(\Omega)} \|f\|_{\mathcal{H}^{2\gamma}}.$$

A.2. **Sobolev Embedding & Sampling Inequalities.**

**Theorem A.1** (Sobolev embedding theorem [2, Thm. 4.12]). *Suppose $\Omega \subset \mathbb{R}^d$ is a bounded set with Lipschitz boundary and that for $\alpha \in \mathbb{N}$ it holds that $\gamma > d/2 + \alpha$. Then $H^\gamma(\Omega)$ is continuously embedded in $C^\alpha(\Omega)$ and it holds that*

$$\|u\|_{C^\alpha(\Omega)} \leqslant C_\Omega \|u\|_{H^\gamma(\Omega)}$$

*for an embedding constant $C_\Omega \geqslant 0$ that depends only on $\Omega$.*

**Theorem A.2** (Sobolev sampling inequality [36, Prop. 3]). *Suppose $\Omega \subset \mathbb{R}^d$ is a bounded set with Lipschitz boundary and consider a set of points $X = \{x_1, \ldots, x_N\} \subset \overline{\Omega}$ with fill distance $h_X := \sup_{x \in \Omega} \inf_{x' \in X} \|x - x'\|_2$. Let $u|_X$ denote the restriction of $u$ to the set $X$. Further consider $\gamma > d/2$ and $0 \leqslant \eta \leqslant \gamma$ and let $u \in H^\gamma(\Omega)$.*

(a) *(Noiseless) Suppose $u|_X = 0$. Then there exists $h_0 > 0$ so that whenever $h_X \leqslant h_0$ we have*

$$\|u\|_{H^\eta(\Omega)} \leqslant C_\Omega h_X^{\gamma - \eta} \|u\|_{H^\gamma(\Omega)},$$

*where $C_\Omega > 0$ is a constant that depends only on $\Omega$.*

(b) *(Noisy) Suppose $u|_X \neq 0$. Then there exists $h_0 > 0$ so that whenever $h_X \leqslant h_0$ we have the inequality*

$$\|u\|_{L^\infty(\Omega)} \leqslant C_\Omega h_X^{\gamma - d/2} \|u\|_{H^\gamma(\Omega)} + 2\|u|_X\|_\infty,$$

*where $C_\Omega > 0$ is a constant that depends only on $\Omega$.*

## Appendix B. Generic CGC framework

As discussed in 4.4.2, for the 1-step method, we implemented a generic code following the computational graph completion (CGC) method introduced in [52]. The process starts by creating an instance of the `ComputationalGraph` class which can be used to build the full graph using the following functions:

- `add_observable`: Takes an observable name (like `"t"` for time) and adds a node for it in the graph. An observable node is considered a root node that doesn't have any incoming edges.
- `add_unknown_fn`: Adds a new node that represents the quantity calculated by the function along with a directed edge form the given source node that new quantity node. It also takes a kernel definition along with its parameters values to build the Gaussian process that will estimate the unknown dependency between the source and target nodes. This Gaussian process can be conditioned on any linear transformation of its outputs, in case no value of the target quantity is observed. This feature allows us to condition the Gaussian process for $H$ here on the $\dot{p}$ and $\dot{q}$, which are the results of a linear transformation (the gradient) of $H$.
- `add_known_fn`: Adds a new node the represents a quantity calculated by a known function that takes another source node quantity.
- `add_aggregator`: Aggregates the quantities of two or more nodes into a single new node that can be used as the source of further computation.
- `add_constraint`: Adds a node that represents a constraint function on a given source node quantity. The output of that constraint function should typically be zero. in an optimized state, hence the value of the constraint functions will be used as a penalty in the optimization process. In the Hamiltonian systems problem here, the constraints function would be the difference between the gradient of $H$ and the values of $\dot{q}$ and $\dot{p}$.

Using these methods, we can transform the 1-step computational graph in figure 1 into the Python code in listing 1 for one-dimensional Hamiltonian systems. The completion process itself

can be done by calling the `complete` method of the `ComputationalGraph` object. This method takes two Numpy arrays:

- `X`: which contains the observed values of all the quantities in the graph and some initialization for the non-observed values. The order of the columns of this matrix should follow the observables order given in the initialization of the `ComputationalGraph` itself.
- `M`: which is a boolean array indicating what values are observed and what values are not.

```python
import jax
import cgc
from cgc.utils import KernelParameter as KP

graph = cgc.ComputationalGraph(observables_order=["t", "p", "q", "H"])

graph.add_observable("t")
graph.add_unknown_fn(
    "t", "q",
    kernel="gaussian", kernel_parameters={"scale": KP(1.0)}
)
graph.add_unknown_fn(
    "t", "p",
    kernel="gaussian", kernel_parameters="scale": KP(1.0)}
)

graph.add_known_fn("p", "p_dot", cgc.grad)
graph.add_known_fn("q", "q_dot", cgc.grad)
graph.add_known_fn("p_dot", "-p_dot", lambda p_dot: -p_dot)

graph.add_aggregator(["q_dot", "-p_dot"], "qp_dot")

graph.add_aggregator(["p", "q"], "pq")
graph.add_unknown_fn(
    "pq", "H",
    linear_functional=jax.jacobian, observations="qp_dot",
    kernel="gaussian", kernel_parameters="scale": KP(1.0)}
)
graph.add_known_fn("H", "grad_H", cgc.grad)

graph.add_aggregator(["q_dot", "grad_H"], "(q_dot, grad_H)")
graph.add_aggregator(["p_dot", "grad_H"], "(p_dot, grad_H)")

def p_dot_constraint(p_dot_grad_H):
    p_dot, grad_H = p_dot_grad_H[:, 0], p_dot_grad_H[:, 1:]
    return p_dot + grad_H[:, 1]

def q_dot_constraint(q_dot_grad_H):
    q_dot, grad_H = q_dot_grad_H[:, 0], q_dot_grad_H[:, 1:]
    return q_dot - grad_H[:, 0]

graph.add_constraint("(p_dot, grad_H)", "W1", p_dot_constraint)
graph.add_constraint("(q_dot, grad_H)", "W2", q_dot_constraint)
```

LISTING 1. Implementation of one-dimensional Hamiltonian computational graph

The `complete` method runs the L-BFGS optimization algorithm on the CGC loss described in [52] and returns the `Z` array which is a copy of the input `X` array with the non-observed values filled in and completed. This new `Z` matrix can be used to retrieve any unknown function in the graph and use it for further inference and computations. We use that to retrieve the Hamiltonian function and externally integrate it with an integrator in the experiments reported in section 5.

**Remark B.1.** *Since the 2-step method provides closed-form solutions for recovering $q, p, H$, or equivalently, in the 2-step computational graph (figure 1), the edges are replaced by closed-form kernel regression solutions, thus there is no need for a `complete` run. Consequently, no optimization is performed for the 2-step method, and computationally, only the nodes of the graph are considered.*
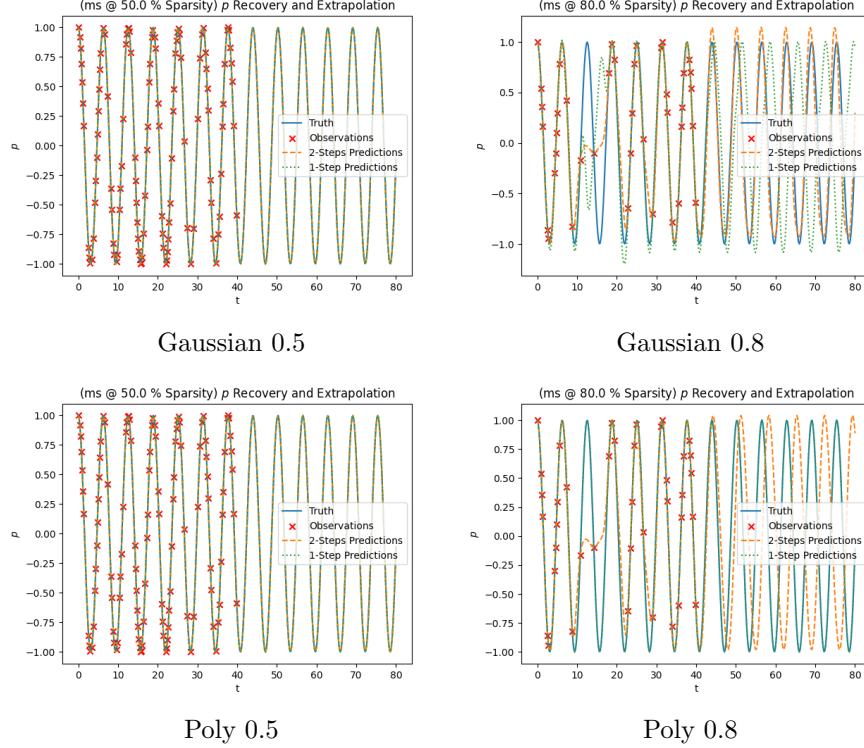
## Appendix C. Supplementary Tables & Plots



Gaussian 0.5

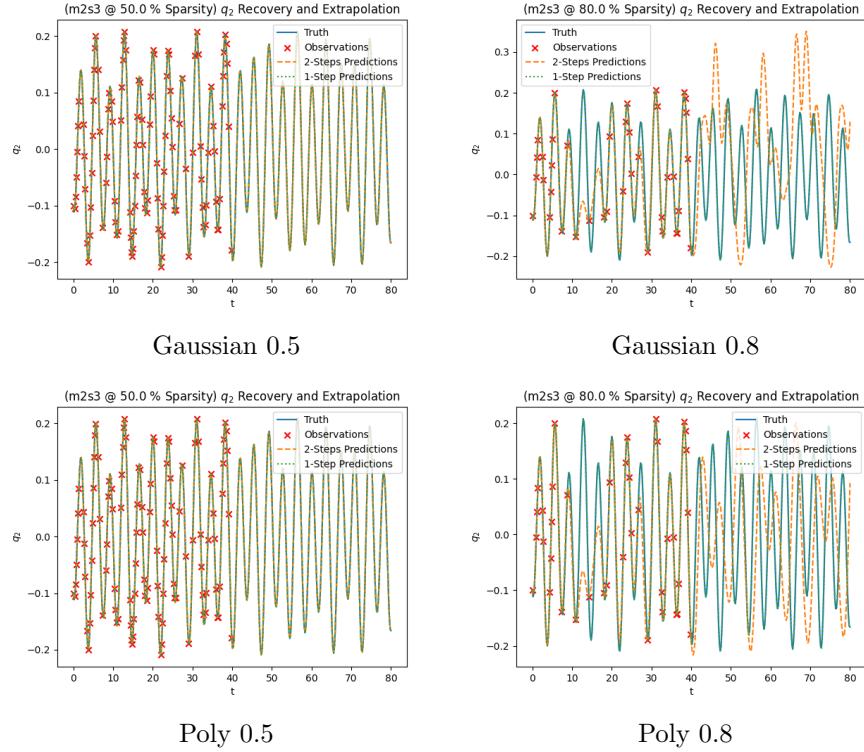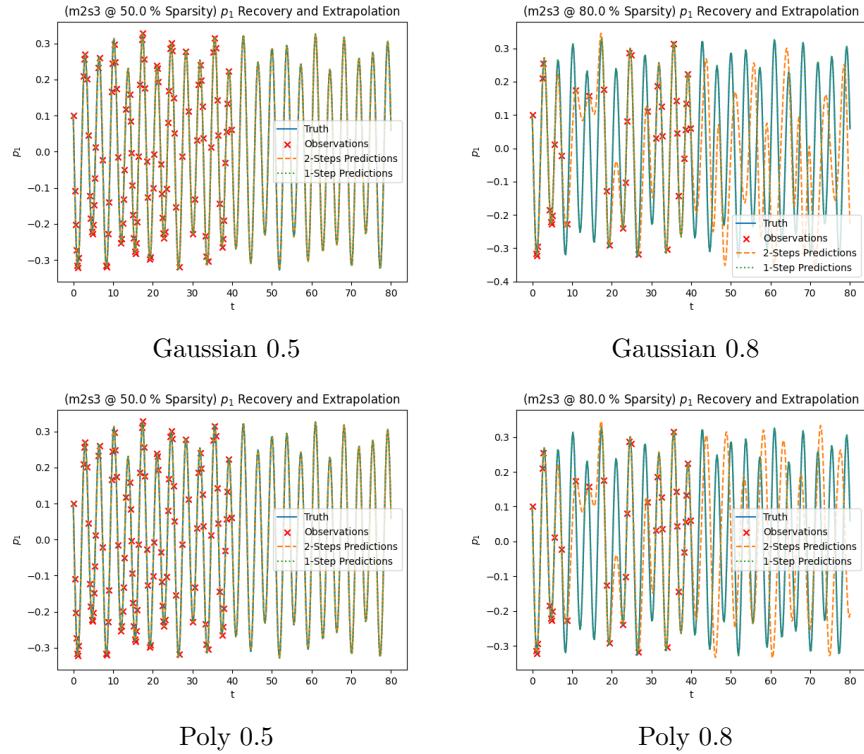Gaussian 0.8

Poly 0.5

Poly 0.8

FIGURE 13. Recovery of $p$ in the Mass-Spring system.

FIGURE 14. Recovery of $q_2$ in the Two-Mass-Three-Spring system.



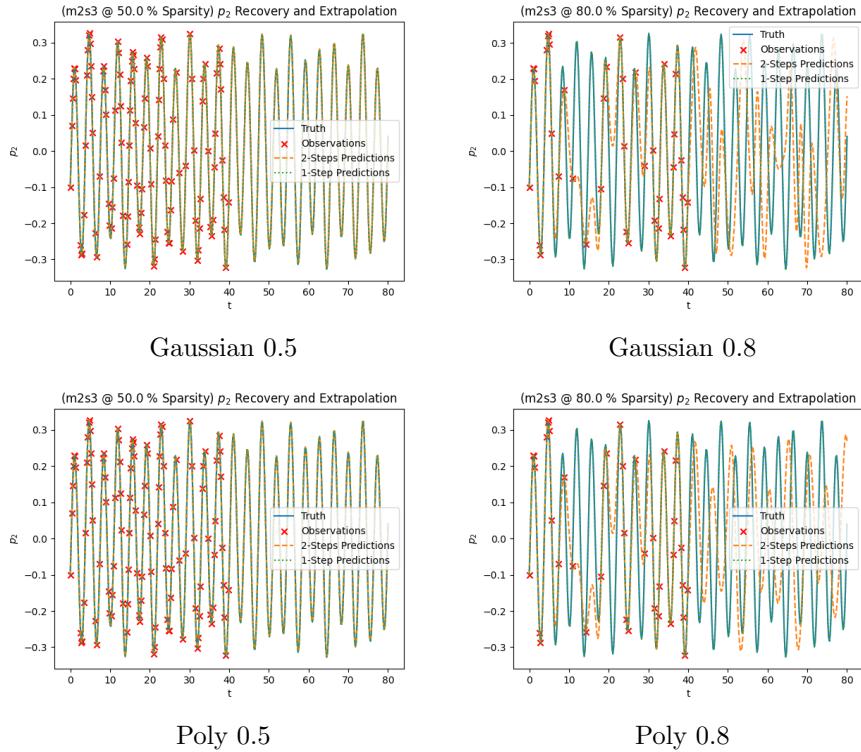FIGURE 15. Recovery of $p_1$ in the Two-Mass-Three-Spring system.

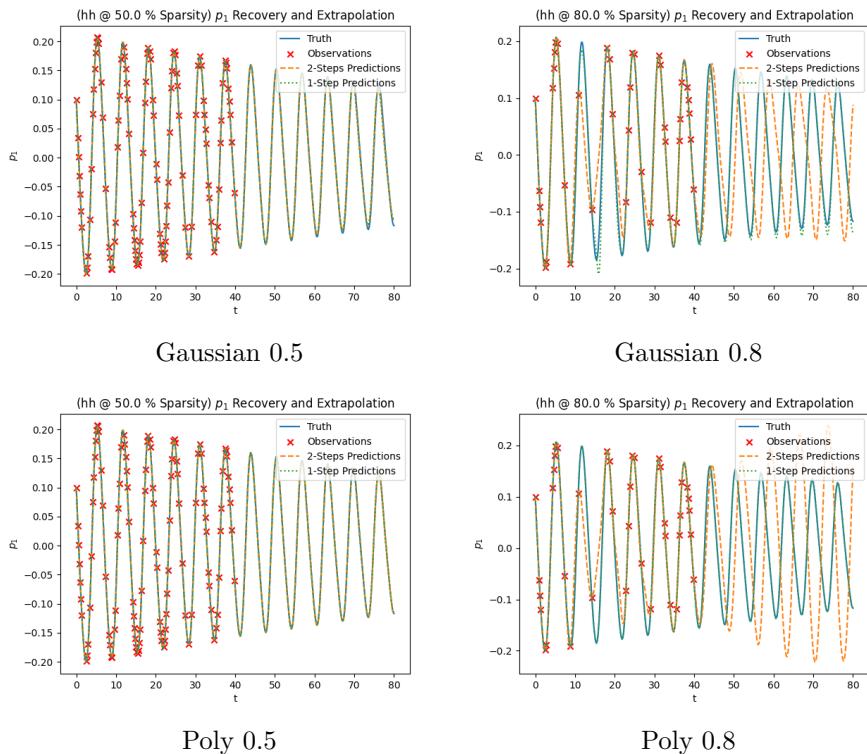FIGURE 16. Recovery of $p_2$ in the Two-Mass-Three-Spring system.



FIGURE 17. Recovery of $p_1$ in the Hénon-Heiles system.

Gaussian 0.5

Gaussian 0.8

Poly 0.5

Poly 0.8

FIGURE 18. Recovery of $p_2$ in the Hénon-Heiles system.



$p$-error, Gaussian

$q$-error, Gaussian

$p$-error, Separable Polynomial

$q$-error, Separable Polynomial

FIGURE 19. Interpolation relative errors for the Mass-Spring System.

$p$-error, Gaussian

$q$-error, Gaussian

$p$-error, Separable Polynomial

$q$-error, Separable Polynomial

FIGURE 20. Interpolation relative errors for the Two-Mass-Three-Spring System.



$p$-error, Gaussian

$q$-error, Gaussian

$p$-error, Separable Polynomial

$q$-error, Separable Polynomial

FIGURE 21. Interpolation relative errors for the Hénon-Heiles System.

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.214 ± 0.000 | 0.003 ± 0.000 | **0.038 ± 0.000** |
| 0.5 | 0.126 ± 0.132 | 1.466 ± 2.201 | **0.019 ± 0.031** | **0.138 ± 0.086** |
| 0.6 | 0.666 ± 0.504 | 8.598 ± 12.837 | **0.153 ± 0.263** | **2.477 ± 4.921** |
| 0.7 | 8.368 ± 6.274 | 58.426 ± 62.178 | **4.585 ± 7.798** | **33.944 ± 65.119** |
| 0.8 | 26.476 ± 15.010 | 102.456 ± 43.744 | **20.859 ± 19.867** | **96.970 ± 59.877** |
| 0.9 | 75.576 ± 15.855 | 134.439 ± 15.315 | **53.391 ± 33.219** | **88.668 ± 55.595** |

Gaussian kernel

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.189 ± 0.000 | 0.003 ± 0.000 | **0.061 ± 0.000** |
| 0.5 | 0.126 ± 0.132 | 1.221 ± 1.085 | **0.009 ± 0.003** | **0.157 ± 0.183** |
| 0.6 | 0.666 ± 0.504 | 3.099 ± 3.629 | **0.019 ± 0.013** | **0.136 ± 0.095** |
| 0.7 | 8.368 ± 6.274 | 16.802 ± 17.617 | **0.037 ± 0.024** | **0.264 ± 0.212** |
| 0.8 | 26.476 ± 15.010 | 76.919 ± 42.785 | **0.095 ± 0.068** | **0.283 ± 0.233** |
| 0.9 | 75.576 ± 15.855 | 145.455 ± 10.461 | **0.069 ± 0.022** | **0.167 ± 0.081** |

Separable polynomial kernel

TABLE 6. Relative errors (mean ± std) for $p$ in the Mass-Spring system.

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.209 ± 0.000 | 0.155 ± 0.000 | **0.314 ± 0.000** |
| 0.5 | 0.334 ± 0.349 | 1.469 ± 1.315 | **0.060 ± 0.054** | **0.242 ± 0.110** |
| 0.6 | 1.553 ± 1.417 | 5.443 ± 7.648 | **0.031 ± 0.011** | **0.196 ± 0.099** |
| 0.7 | 16.758 ± 10.896 | 55.543 ± 58.337 | **2.077 ± 5.072** | **3.072 ± 6.465** |
| 0.8 | 42.585 ± 18.315 | 137.835 ± 50.347 | **6.504 ± 8.376** | **13.412 ± 18.145** |
| 0.9 | 103.994 ± 12.912 | 154.808 ± 24.992 | **48.381 ± 28.216** | **74.761 ± 34.479** |

Gaussian kernel

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 0.140 ± 0.000 | 0.004 ± 0.000 | **0.127 ± 0.000** |
| 0.5 | 0.334 ± 0.349 | 1.549 ± 1.572 | **0.017 ± 0.006** | **0.183 ± 0.143** |
| 0.6 | 1.553 ± 1.417 | 5.830 ± 8.598 | **0.042 ± 0.046** | **0.272 ± 0.231** |
| 0.7 | 16.758 ± 10.896 | 71.835 ± 58.325 | **0.084 ± 0.051** | **0.345 ± 0.195** |
| 0.8 | 42.585 ± 18.315 | 137.481 ± 37.661 | **0.145 ± 0.091** | **0.399 ± 0.245** |

Separable polynomial kernel

TABLE 7. Relative errors (mean ± std) for $p$ in the Two-Mass-Three-Spring system.

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 2.882 ± 0.000 | 0.002 ± 0.000 | **2.892 ± 0.000** |
| 0.5 | 0.306 ± 0.277 | 2.764 ± 0.369 | **0.030 ± 0.024** | 3.017 ± 0.150 |
| 0.6 | 0.995 ± 0.813 | 3.571 ± 1.597 | **0.054 ± 0.033** | 3.001 ± 0.328 |
| 0.7 | 9.745 ± 8.983 | 17.634 ± 15.158 | **2.255 ± 5.360** | **8.057 ± 9.954** |
| 0.8 | 25.900 ± 13.807 | 65.628 ± 42.015 | **6.284 ± 8.435** | **21.636 ± 27.447** |
| 0.9 | 77.538 ± 19.790 | 139.084 ± 15.678 | **24.388 ± 18.010** | **43.922 ± 29.839** |

Gaussian kernel

| Sparsity | Two-Steps Method | | One-Step Method | |
|---|---|---|---|---|
| | Interpolation | Extrapolation | Interpolation | Extrapolation |
| 0.0 | **0.000 ± 0.000** | 1.149 ± 0.000 | 0.002 ± 0.000 | **1.149 ± 0.000** |
| 0.5 | 0.306 ± 0.277 | 1.766 ± 0.712 | **0.025 ± 0.017** | **1.251 ± 0.104** |
| 0.6 | 0.995 ± 0.813 | 3.100 ± 2.452 | **0.046 ± 0.041** | **1.197 ± 0.054** |
| 0.7 | 9.745 ± 8.983 | 17.621 ± 18.179 | **0.061 ± 0.040** | **1.355 ± 0.191** |
| 0.8 | 25.900 ± 13.807 | 76.579 ± 44.581 | **0.106 ± 0.063** | **1.425 ± 0.203** |
| 0.9 | 77.538 ± 19.790 | 144.148 ± 13.160 | **0.146 ± 0.094** | **1.822 ± 0.503** |

Separable polynomial kernel

TABLE 8.  Relative errors (mean ± std) for $p$ in the Hénon-Heiles system.



$p$-error, Gaussian

$q$-error, Gaussian

$p$-error, Separable Polynomial

$q$-error, Separable Polynomial

FIGURE 22.  Interpolation relative errors for the Nonlinear Pendulum System.