Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting

Yann Claes[1*], Vân Anh Huynh-Thu[1] and Pierre Geurts[1]

[1*]Montefiore Institute, University of Liège, Liège, 4000, Belgium.

*Corresponding author(s). E-mail(s): y.claes@uliege.be;
Contributing authors: vahuynh@uliege.be; p.geurts@uliege.be;

Abstract

Learning processes by exploiting restricted domain knowledge is an important task across a plethora of scientific areas, with more and more hybrid training methods additively combining data-driven and model-based approaches. Although the obtained models are more accurate than purely data-driven models, the optimization process usually comes with sensitive regularization constraints. Furthermore, while such hybrid methods have been tested in various scientific applications, they have been mostly tested on dynamical systems, with only limited study about the influence of each model component on global performance and parameter identification. In this work, we introduce a new hybrid training approach based on partial dependence, which removes the need for intricate regularization. Moreover, we assess the performance of hybrid modeling against traditional machine learning methods on standard regression problems. We compare, on both synthetic and real regression problems, several approaches for training such hybrid models. We focus on hybrid methods that additively combine a parametric term with a machine learning term and investigate model-agnostic training procedures. Therefore, experiments are carried out with different types of machine learning models, including tree-based models and artificial neural networks. We also extend our partial dependence optimization process for dynamical systems forecasting and compare it to existing schemes.*

Keywords: Physics-informed machine learning, Physics-guided machine learning, Supervised regression, Tree-based methods, Neural networks, Partial dependence, Hybrid modeling.

---

*Our Python implementations of the hybrid methods are available at https://github.com/yannclaes/kg-regression.

1 Introduction

For the past decades, machine learning (ML) models have been developed to tackle a variety of real-life problems, complementing/replacing model-based (MB) approaches, which mostly remain approximations that make stringent assumptions about the system under study. Traditional ML approaches are said to be *data-driven*, i.e. their prediction model is solely built from some learning dataset. While their design comes with great expressiveness, they are likely to be subject to over-fitting without enough training examples and to show a lack of robustness on unseen samples, with predictions that can be inconsistent with respect to domain knowledge (Daw et al., 2022; De Bézenac et al., 2019). To overcome this generalization issue, hybrid approaches have been introduced to incorporate *a priori* domain knowledge within statistical models, which can be leveraged in a multitude of ways (see Von Rueden et al., 2021; Karniadakis et al., 2021; Willard et al., 2022 for reviews). A popular hybrid approach considers an additive combination of MB and ML components, and the success of these methods have been shown empirically on a range of synthetic and real-world problems (Yin et al., 2021; Mehta et al., 2021; Donà et al., 2022). However, a significant challenge with such methods is to find the right balance between MB and ML components, if one is interested in retrieving a meaningful estimation of the parametric prior. This generally requires the introduction of regularization terms in the loss function, whose weighting coefficients are complex to optimize. Moreover, while these hybrid optimization methods have been mostly applied to dynamical systems, they have not been thoroughly studied in the context of standard regression problems, and the majority of ML models considered in these approaches are neural networks, leaving aside other methods. Our contributions are the following:

- We introduce a new hybrid training method based on partial dependence (PD), which avoids regularization constraints and makes fewer assumptions than other approaches.
- We first investigate empirically the performance and benefits of hybrid methods against data-driven methods on *static* regression problems (in contrast to dynamical problems). The static context removes a layer of complexity related to the temporal correlation between observed states, which makes it easier to assess the interaction between the MB and ML components. We compare different approaches for training such hybrid additive models, including our PD-based training. We highlight specific assumptions under which these approaches are expected to work well and relate their differences in terms of prediction and parameter recovery performance. We focus on model-agnostic approaches, where the ML term can be of any type, and we compare tree-based methods against neural networks. Tree-based methods have several advantages over neural networks, which motivate their use on static regression problems: they have much less hyper-parameters to tune, appear robust to the presence of irrelevant features and have been shown to often outperform neural networks on tabular data (Grinsztajn et al., 2022).
- We extend our partial-dependence-based optimization scheme to dynamical systems forecasting and compare it to the other hybrid training approaches. We show that it

performs equally well and that it produces parametric physical terms of very good quality compared to other hybrid additive training schemes.

Note: This paper is an extended version of (Claes et al., 2023). In addition to improved descriptions of the algorithms and experiments, this paper includes the adaptation and evaluation of the methods for dynamical systems forecasting, while (Claes et al., 2023) focused exclusively on supervised regression.

2 Related work

Hybrid additive modeling methods emerged several decades ago, combining first-principles models with different ML models. Already in the 1990's, approaches derived by Kramer et al. (1992); Johansen and Foss (1992); Thompson and Kramer (1994) complemented physics-based models with neural networks, weighting contributions of both components (e.g. through radial basis function networks), to achieve enhanced physical consistency with better generalization properties. More recently, other works applied the same principles to model dynamical systems in various domains, still massively relying on neural networks (Yin et al., 2021; Qian et al., 2021; Takeishi and Kalousis, 2021; Wehenkel et al., 2022). In a more standard regression setting, Chen et al. (2007); Yu et al. (2010); Zhang et al. (2019) combined a linear parametric term with a tree-based ML term.

On the challenge of finding the right balance between the MB and ML components, previous works have introduced regularization of the ML term to reduce parameter identification issues in the decomposition (Wouwer et al., 2004; Hu et al., 2011). Further works on this matter introduced constraints in the learning objective to better control contributions of the MB/ML components (Yin et al., 2021; Donà et al., 2022). Elements of discussion about the well-posedness of this additive decomposition have also been introduced in previous works: Yin et al. (2021) showed the existence and uniqueness of an optimal MB-ML pair when the contributions of the statistical model are constrained to be minimal, and Donà et al. (2022) demonstrated the convergence of an algorithm alternating between the optimization of both parts, without however any guarantee about convergence points. Nevertheless, most of these works only addressed dynamical systems forecasting.

3 Static supervised regression

In this section, we focus first on hybrid additive modeling for standard supervised regression. We formally define the problem of hybrid additive modeling in Section 3.1 and then describe various methods to address it in Section 3.2, including a new proposal based on partial dependence (Section 3.2.3). All methods are compared on several synthetic and real problems in Section 3.3.

3.1 Problem statement

Let us define a regression problem, where $y \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^d$, with $d \in \mathbb{N}_+$, are drawn from a distribution $p(\mathbf{x}, y)$ such that $y = f(\mathbf{x}) + \varepsilon$ with $f : \mathbb{R}^d \mapsto \mathbb{R}$ the partially known

generating function and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ the noise term. We focus on problems such that $f(\mathbf{x})$ can be decomposed as:

Assumption 1 (A1, Additivity) $y = f_k(\mathbf{x}_k) + f_a(\mathbf{x}) + \varepsilon$, where $\mathbf{x}_k$ is a subset of $K \leq d$ input variables.

We assume partial knowledge of the generating function through some known algebraic function $h_k^{\theta_k}(\mathbf{x}_k) \in \mathcal{H}_k$ with tunable parameters θ_k, such that for the optimal parameter values θ_k^* we have $h_k^{\theta_k^*} = f_k$. The residual term $f_a(\mathbf{x})$ is unknown and is approximated in this work through an ML component $h_a^{\theta_a} \in \mathcal{H}_a$, with parameters θ_a[1]. The final model $h \in \mathcal{H}$ is denoted $h(\mathbf{x}) = h_k^{\theta_k}(\mathbf{x}_k) + h_a^{\theta_a}(\mathbf{x})$, with the function space $\mathcal{H}$ defined as $\mathcal{H}_k + \mathcal{H}_a$. A1 is common when MB methods and ML models are combined (Takeishi and Kalousis, 2021; Yin et al., 2021; Donà et al., 2022; Wehenkel et al., 2022).

Given a learning sample of N input-output pairs $LS = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, drawn from $p(\mathbf{x}, y)$, we seek to identify a function $h = h_k^{\theta_k} + h_a^{\theta_a}$, i.e. parameters θ_k and θ_a, that minimizes the following two distances:

$$d(h, y) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)}\{(h(\mathbf{x}) - y)^2\}, \tag{1}$$

$$d_k(h_k^{\theta_k}, f_k) = \mathbb{E}_{\mathbf{x}_k \sim p(\mathbf{x}_k)}\{(h_k^{\theta_k}(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2\}. \tag{2}$$

The first distance measures the standard generalization error of the global model h. The hope is that taking h_k into account will help learning a better global model than fitting directly a pure data-driven model on y, especially in the small sample size regime. The second distance d_k measures how well the tuned h_k approximates f_k. The main motivation for this second objective is interpretability: one expects that the algebraic form of h_k will be derived from first principles by domain experts, who will be interested in estimating the parameters of this term from data. An alternative to d_k is a loss that would compare the estimated and optimal parameters $\hat{\theta}_k$ and θ_k^* (e.g., $||\hat{\theta}_k - \theta_k^*||^2$). d_k however has the advantage not to require θ_k^* to be fully identifiable, i.e. there can exist several sets of parameters θ_k^* such that $h_k^{\theta_k^*} = f_k$.

The following approximation of (1) can be used as training objective:

$$\hat{d}(h, y; LS) = \frac{1}{N} \sum_{i=1}^N (h(\mathbf{x}_i) - y_i)^2. \tag{3}$$

Minimizing the distance in (2) is expected to be challenging and sometimes even ill-posed. Indeed, if h_a is too powerful, it could capture f entirely and leave little room for the estimation of f_k. Finding the right balance between h_k and h_a is thus very challenging, if not impossible, using only guidance of the learning sample LS. Unlike (1), (2) cannot be estimated from a sample of input-output pairs and hence cannot be explicitly used to guide model training. There are however several scenarios that

---

[1]In the following, $h_k^{\theta_k}$ and $h_a^{\theta_a}$ will sometimes be denoted simply as h_k and h_a to lighten the notations.

can make the problem easier. In the following, we will discuss the optimality of hybrid methods under two additional assumptions:

Assumption 2 (A2, Disjoint features) Let $\mathbf{x}_a$ be a subset of features disjoint from $\mathbf{x}_k$ ($\mathbf{x}_k \cap \mathbf{x}_a = \emptyset$). There exists a function $f_a^r(\mathbf{x}_a)$ such that $f_a(\mathbf{x}) = f_a^r(\mathbf{x}_a)$ for all $\mathbf{x}$.

Assumption 3 (A3, Independence) Features in $\mathbf{x}_k$ are independent from features in $\mathbf{x}_a$ ($\mathbf{x}_k \perp\!\!\!\perp \mathbf{x}_a$).

Problems where A2 is satisfied are easier as f_k captures all the dependence of y on $\mathbf{x}_k$. Furthermore, in problems where A3 is not met, it might be hard to distinguish real contributions from $\mathbf{x}_k$ to f from those due to correlations with features not in $\mathbf{x}_k$. Note that these assumptions do not strictly define our modeling framework, but are rather used in further developments and experiments as tools to derive claims about the optimality and convergence of hybrid training methods on problems that do (or do not) satisfy them.

3.2 Training methods

We focus on model-agnostic approaches, i.e. that can be applied with any algebraic function h_k and any type of ML model h_a. For both terms, we only assume access to training functions, respectively denoted fit^{h_k}, $\text{fit}^{h_k+\gamma}$, and fit^{h_a}, that can estimate each model parameters, respectively θ_k, (θ_k, γ) and θ_a, so as to minimize the mean squared error (MSE) over LS (see below for the meaning of γ), where parametric methods rely on gradient descent. Pseudo-codes of methods in Sections 3.2.1 and 3.2.2 are given in Appendix E.

3.2.1 Sequential training of h_k and h_a

This baseline approach first fits h_k on the observed output y, then fits h_a on the resulting residuals, as done by Zhang et al. (2019) (see Algorithm 3 in Appendix E). More precisely, we first train $h_k^{\theta_k}$ on y by introducing a constant term $\gamma \in \mathbb{R}$, such that

$$(\hat{\theta}_k, \hat{\gamma}) = \text{fit}^{h_k+\gamma}(LS). \tag{4}$$

Our motivation for introducing the term γ will be explained below. Afterwards, we fit h_a on the output residuals: $\hat{\theta}_a = \text{fit}^{h_a}\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k}(\mathbf{x}_i) - \hat{\gamma})\}_{i=1}^N$.

Let $\hat{\mathcal{F}}_k$ be the set of all functions $\hat{f}_k$ mapping $\mathbf{x}_k \in \mathcal{X}_k$ to some value $y \in \mathbb{R}$, i.e. $\hat{\mathcal{F}}_k = \{\hat{f}_k : \mathcal{X}_k \mapsto \mathbb{R}\}$. Under A2 and A3, it can be shown that $\hat{f}_k^* = \arg\min_{\hat{f}_k \in \hat{\mathcal{F}}_k} d(\hat{f}_k, y)$ is such that $\hat{f}_k^*(\mathbf{x}_k) = f_k(\mathbf{x}_k) + C$, for every $\mathbf{x}_k \in \mathcal{X}_k$, with $C = \mathbb{E}_{\mathbf{x}_a}\{f_a^r(\mathbf{x}_a)\}$ (for more details, see Appendix A). Hence, this approach is sound at least asymptotically and justifies the introduction of γ. Note however that even under A2 and A3, we have no guarantee that this approach produces the best estimator for a finite sample size, as $f_a^r(\mathbf{x}_a) + \epsilon$ acts as a pure additive noise term that needs to be averaged out during training. The approaches described in Sections 3.2.2 and 3.2.3 try to overcome this

issue by fitting $h_k^{\theta_k}$ on corrected outputs that are expected to be closer to $f_k(\mathbf{x}_k)$. Regarding sequential optimization, without A2 and A3, the quality of the estimation $h_k^{\hat{\theta}_k}$, according to (2), is not guaranteed as there are problems satisfying A1 such that:

$$\nexists \gamma \in \mathbb{R} : \arg\min_{\hat{f}_k \in \hat{\mathcal{F}}_k} d(\hat{f}_k, y) = f_k + \gamma. \tag{5}$$

An example of such problems will be given in Section 3.3.2.

3.2.2 Alternate training of h_k and h_a

A hybrid additive approach was proposed by Donà et al. (2022), tackling the problem of dynamical systems forecasting, which alternates between updating h_k and updating h_a, using neural networks for h_a. They also introduce specific regularization criteria to try and find the optimal balance between both parts of the model, but we decided to focus on the non-regularized version of the algorithm, as we did not manage to significantly improve the results in the static experiments by adding the regularization process (such process will however have an impact in the dynamical setting). Such alternate training was also proposed by Chen et al. (2007) and Yu et al. (2010) with a single decision tree as h_a and a linear h_k. We include this approach in our comparison, but also investigate it with random forests (Breiman, 2001) and tree gradient boosting (Friedman, 2002).

The optimization process defined by Donà et al. (2022) is as follows: $\hat{\theta}_k$ is firstly initialized by (fully) fitting $h_k^{\theta_k} + \gamma$ on y. Then, we alternate between: (1) a single epoch of gradient descent on $h_k^{\theta_k} + \gamma$ and (2) either a single epoch for h_a (in the case of neural networks, as performed by Donà et al. (2022)) or a complete fit of h_a (in the case of tree-based models). This process is described in Algorithm 5 (Appendix E).

While some theoretical results are given by Donà et al. (2022), convergence of the alternate method towards the optimal solution is not guaranteed in general. Despite an initialization favoring h_k, it is unclear whether a too expressive h_a will not dominate h_k and finding the right balance between these two terms, e.g. by regularizing further h_a, is challenging. Under A2 and A3 however, the population version[2] of the algorithm produces an optimal solution. Indeed, h_k will be initialized as the true f_k, as shown previously, making the residuals $y - h_k$ at the first iteration, as well as h_a, independent of $\mathbf{x}_k$. h_k will thus remain unchanged (and optimal) at subsequent iterations.

3.2.3 Partial Dependence-based training of h_k and h_a

We propose a novel approach relying on partial dependence (PD) functions (Friedman, 2001) to produce a proxy dataset depending only on $\mathbf{x}_k$ to fit h_k. PD measures how a given subset of features impact the prediction of a model, on average. Let $\mathbf{x}_k$ be the subset of interest and $\mathbf{x}_{-k}$ its complement, with $\mathbf{x}_k \cup \mathbf{x}_{-k} = \mathbf{x}$, then the PD of a

---

[2]i.e., assuming an infinite training sample size and consistent estimators.

function $f(\mathbf{x})$ on $\mathbf{x}_k$ is:

$$PD(f, \mathbf{x}_k) = \mathbb{E}_{\mathbf{x}_{-k}}\left[f(\mathbf{x}_k, \mathbf{x}_{-k})\right] = \int f(\mathbf{x}_k, \mathbf{x}_{-k}) p(\mathbf{x}_{-k}) d\mathbf{x}_{-k}, \tag{6}$$

where $p(\mathbf{x}_{-k})$ is the marginal distribution of $\mathbf{x}_{-k}$. Under A1 and A2, the PD of $f(\mathbf{x}) = f_k(\mathbf{x}_k) + f_a^r(\mathbf{x}_a)$ is (Friedman, 2001):

$$PD(f, \mathbf{x}_k) = f_k(\mathbf{x}_k) + C, \text{ with } C = E_{\mathbf{x}_a}\{f_a^r(\mathbf{x}_a)\}. \tag{7}$$

The idea of our method is to first fit any sufficiently expressive ML model $h_a(\mathbf{x})$ on LS and to compute its PD w.r.t. $\mathbf{x}_k$ to obtain a first approximation of $f_k(\mathbf{x}_k)$ (up to a constant). Although computing the actual PD using (6) requires in principle access to the input distribution, an approximation can be estimated from LS as follows:

$$\widehat{PD}(h_a, \mathbf{x}_k; LS) = \frac{1}{N} \sum_{i=1}^{N} h_a(\mathbf{x}_k, \mathbf{x}_{i,-k}), \tag{8}$$

where $\mathbf{x}_{i,-k}$ denotes the values of $\mathbf{x}_{-k}$ in the i-th sample of LS. A new dataset of pairs $(\mathbf{x}_k, \widehat{PD}(h_a, \mathbf{x}_k; LS))$ can then be built to fit h_k. In our experiments, we consider only the $\mathbf{x}_k$ values observed in the learning sample but $\widehat{PD}(h_a, \mathbf{x}_k; LS)$ could also be estimated at other points $\mathbf{x}_k$ to artificially increase the size of the proxy dataset.

In practice, optimizing θ_k only once on the PD of h_a could leave residual dependence of $\mathbf{x}_k$ on the resulting $y - h_k^{\hat{\theta}_k}(\mathbf{x}_k) - \hat{\gamma}$. We thus repeat the sequence of fitting h_a on the latter residuals, then fitting h_k on the obtained $\widehat{PD}(h_a^{\hat{\theta}_a}, \mathbf{x}_k; LS) + h_k^{\hat{\theta}_k}(\mathbf{x}_k) + \hat{\gamma}$, with $\hat{\theta}_k$, $\hat{\gamma}$ and $\hat{\theta}_a$ the current optimized parameter vectors (see Algorithm 1).

Algorithm 1 Partial Dependence Optimization - Static problems

Input: $LS = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
$\hat{\theta}_a^{(0)} \leftarrow \text{fit}^{h_a}(LS)$
$(\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}) \leftarrow \text{fit}^{h_k + \gamma}(\{(\mathbf{x}_{k,i}, \widehat{PD}(h_a^{\hat{\theta}_a^{(0)}}, \mathbf{x}_{k,i}; LS))\}_{i=1}^{N})$
for $n = 1$ **to** N_r **do**
$\quad \hat{\theta}_a^{(n)} \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, \ y_i - h_k^{\hat{\theta}_k^{(n-1)}}(\mathbf{x}_{k,i}) - \hat{\gamma}^{(n-1)})\}_{i=1}^{N})$
$\quad (\hat{\theta}_k^{(n)}, \hat{\gamma}^{(n)}) \leftarrow \text{fit}^{h_k + \gamma}(\{(\mathbf{x}_{k,i}, h_k^{\hat{\theta}_k^{(n-1)}}(\mathbf{x}_{k,i}) + \hat{\gamma}^{(n-1)} + \widehat{PD}(h_a^{\hat{\theta}_a^{(n)}}, \mathbf{x}_{k,i}; LS))\}_{i=1}^{N})$
end for
$\hat{\theta}_a^{(N_r+1)} \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, \ y_i - h_k^{\hat{\theta}_k^{(N_r)}}(\mathbf{x}_{k,i}) - \hat{\gamma}^{(N_r)})\}_{i=1}^{N})$
return $h_k^{\hat{\theta}_k^{(N_r)}} + \hat{\gamma}^{(N_r)} + h_a^{\hat{\theta}_a^{(N_r+1)}}$

The main advantage of this approach over the alternate one is to avoid domination of h_a over h_k. Unlike the two previous approaches, this one is also sound even if A3 is not satisfied as it is not a requirement for (7) to hold. One drawback is that it requires

h_a to capture well the dependence of f on $\mathbf{x}_k$ so that its PD is a good approximation of f_k. The hope is that even if it is not the case at the first iteration, fitting h_k, that contains the right inductive bias, will make the estimates better and better over the iterations.

3.3 Experiments

We compare the different methods on several regression datasets, both simulated and real. Performance is measured through estimates of (1) and (2) (the latter only on simulated problems) on a test set TS, respectively denoted $\hat{d}(h, y; TS)$ and $\hat{d}_k(h_k^{\theta_k}, f_k; TS)$. In some cases, we also report $\mathrm{rMAE}(\theta_k^*, \theta_k)$, the relative mean absolute error between θ_k^* and θ_k (lower is better for all measures). For hybrid approaches, we use as h_a either a multilayer perceptron (MLP), gradient boosting with decision trees (GB) or random forests (RF). We compare these hybrid models to a standard data-driven model that uses only h_a. We also compare fitting h_a with and without input filtering, i.e. respectively removing or keeping $\mathbf{x}_k$ from its inputs, to verify convergence claims about h_k in Section 3.2.2, under A2. Architectures (e.g. for MLP, the number of layers and neurons) are kept fixed across training methods to allow a fair comparison between them, and are given in Appendix B. We monitor (3) on a validation set and select the model that reaches the lowest validation loss.

3.3.1 Independent input features (A2 and A3 satisfied)

Friedman problem

We consider the following synthetic regression problem:

$$y = \theta_0 \sin(\theta_1 x_0 x_1) + \theta_2 (x_2 - \theta_3)^2 + \theta_4 x_3 + \theta_5 x_4 + \sum_{j=5}^{9} 0 x_j + \varepsilon,$$

where $x_j \sim \mathcal{U}(0, 1), j = 0, \ldots 9$, and $\varepsilon \sim \mathcal{N}(0, 1)$ (Friedman et al., 1983). We generate 10 different datasets using 10 different sets of values for $\theta_0, \ldots, \theta_5$, each with 300, 300 and 600 samples for respectively the training, validation and test sets. For the hybrid approaches, we use the first term as prior knowledge, i.e. $f_k(\mathbf{x}_k) = \theta_0 \sin(\theta_1 x_0 x_1)$.

We see in Table 1 that all hybrid training schemes outperform their data-driven counterpart. They come very close to the ideal $f_k \to h_a$ method, and sometimes even slightly better, probably due to chance. Sequential fitting of h_k and h_a performs as well as the alternate or PD-based approaches, as A2 and A3 are satisfied for this problem (see Section 3.2.1). Filtering generally improves the performance of hybrid schemes as A2 is satisfied. PD-based optimization yields good approximations of f_k (as shown by a low $\hat{d}_k$). The alternate approach follows closely whereas the sequential one ends up last, which can be expected as fitting h_k only on y induces a higher noise level centered around $\mathbb{E}_{\mathbf{x}_a} \{f_a(\mathbf{x}_a)\}$, while the other approaches benefit from reduced perturbations through h_a estimation, as explained in Section 3.2.1. Filtering vastly decreases $\hat{d}_k$ for alternate approaches, supporting claims introduced in Section 3.2.2, while this measure remains unimpaired for sequential and PD-based training by construction.

Table 1 Results on the *Friedman* problem. We report the mean and standard deviation of $\hat{d}$ and $\hat{d}_k$ over the 10 test sets (TS). "$f_k \to h_a$" fits h_a on $y - f_k(\mathbf{x}_k)$. "Unfiltered" indicates that all the features are used as inputs of h_a, while "Filtered" indicates that the features $\mathbf{x}_k$ are removed from the inputs of h_a.

		$\hat{d}(h, y; TS)$		$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$	
	Method	Unfiltered	Filtered	Unfiltered	Filtered
	$f_k \to h_a$	1.58 ± 0.33	1.23 ± 0.10	-	
	Sequential	1.54 ± 0.31	1.43 ± 0.13	0.18 ± 0.16	
MLP	Alternate	1.43 ± 0.09	1.32 ± 0.09	0.10 ± 0.09	0.02 ± 0.02
	PD-based	1.54 ± 0.12	1.38 ± 0.09	0.06 ± 0.07	
	h_a only	2.62 ± 0.75		-	
	$f_k \to h_a$	1.73 ± 0.09	1.75 ± 0.12	-	
	Sequential	1.74 ± 0.11	1.81 ± 0.14	0.18 ± 0.16	
GB	Alternate	1.79 ± 0.11	1.78 ± 0.15	0.91 ± 1.45	0.06 ± 0.06
	PD-based	1.77 ± 0.13	1.78 ± 0.12	0.03 ± 0.02	
	h_a only	3.43 ± 0.94		-	
	$f_k \to h_a$	2.03 ± 0.18	1.96 ± 0.17	-	
	Sequential	2.11 ± 0.23	2.05 ± 0.24	0.18 ± 0.16	
RF	Alternate	2.03 ± 0.19	1.98 ± 0.17	0.04 ± 0.03	0.04 ± 0.04
	PD-based	2.16 ± 0.27	2.09 ± 0.26	0.16 ± 0.15	
	h_a only	5.58 ± 1.91		-	

3.3.2 Correlated input features (A3 not satisfied)

Correlated Linear problem.

Let $y = \beta_0 x_0 + \beta_1 x_1 + \varepsilon$, with $\beta_0 = -0.5, \beta_1 = 1, \mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, and $\varepsilon \sim \mathcal{N}(0, 0.5^2)$. We generate 50, 50 and 600 samples respectively for the training, validation and test sets. We use as known term $f_k(\mathbf{x}_k) = \beta_0 x_0$. Regressing y on x_0 yields the least-squares solution (Greene, 2018):

$$\mathbb{E}\left[\hat{\beta}_0\right] = \beta_0 + \frac{\text{cov}(x_0, x_1)}{\text{var}(x_0)}\beta_1. \tag{9}$$

We set $\text{cov}(x_0, x_1) = 2.25$ and $\text{var}(x_0) = 2$ so that (9) reverses the sign of β_0 and (5) is satisfied. The sequential approach should hence yield parameter estimates of β_0 close to (9) while we expect the others to correct for this bias.

From Table 2, we observe that, contrary to PD-based training, the sequential and alternate methods return very bad estimations of β_0, as A3 is no longer satisfied. Filtering corrects the bias for the alternate approach but degrades the MSE performance for sequential training as it removes the ability to compensate for the h_k misfit.

Correlated Friedman problem.

The structure is identical to the one in Section 3.3.1 but with correlated inputs drawn from a multivariate normal distribution where $\mu_i = 0.5$ and $\text{var}(x_i) = 0.75, \forall i$, and $\text{cov}(x_i, x_j) = \pm 0.3, \forall i \neq j$ (the covariance sign being chosen randomly). Sizes of the

Table 2 Results for the *Correlated Linear* problem. We report $\hat{d}$ and rMAE$(\beta_0^*, \hat{\beta}_0)$, over 10 different datasets.

	Method	$\hat{d}(h,y;TS)$ Unfiltered	Filtered	rMAE$(\beta_0^*, \hat{\beta}_0)$ Unfiltered	Filtered
MLP	Sequential	0.30 ± 0.03	0.74 ± 0.09	224.14 ± 13.48	
	Alternate	0.30 ± 0.02	0.31 ± 0.04	186.65 ± 21.31	15.53 ± 13.57
	PD-based	0.30 ± 0.03	0.29 ± 0.02	26.47 ± 17.32	
GB	Sequential	0.59 ± 0.06	1.38 ± 0.11	224.14 ± 13.48	
	Alternate	0.57 ± 0.06	0.60 ± 0.09	148.75 ± 67.35	24.58 ± 12.20
	PD-based	0.56 ± 0.05	0.64 ± 0.13	36.05 ± 17.50	
RF	Sequential	0.53 ± 0.05	0.90 ± 0.07	224.14 ± 13.48	
	Alternate	0.43 ± 0.04	0.42 ± 0.04	111.04 ± 52.78	45.38 ± 22.39
	PD-based	0.41 ± 0.03	0.43 ± 0.04	57.47 ± 15.55	

Table 3 Results for the *Correlated Friedman* problem.

	Method	$\hat{d}(h,y;TS)$ Unfiltered	Filtered	$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$ Unfiltered	Filtered
MLP	$f_k \to h_a$	1.64 ± 0.23	1.51 ± 0.17	-	
	Sequential	2.07 ± 0.40	2.68 ± 1.38	1.35 ± 1.42	
	Alternate	1.95 ± 0.33	1.62 ± 0.24	0.49 ± 0.44	0.14 ± 0.19
	PD-based	2.24 ± 0.31	1.78 ± 0.30	0.17 ± 0.23	
	h_a only	2.77 ± 0.73		-	
GB	$f_k \to h_a$	2.58 ± 0.45	2.53 ± 0.44	-	
	Sequential	2.90 ± 0.39	3.91 ± 1.49	1.35 ± 1.42	
	Alternating	2.67 ± 0.38	2.62 ± 0.43	0.51 ± 0.53	0.22 ± 0.25
	PD-based	2.54 ± 0.35	2.47 ± 0.36	0.03 ± 0.02	
	h_a only	4.49 ± 0.66		-	
RF	$f_k \to h_a$	3.02 ± 0.45	2.93 ± 0.45	-	
	Sequential	3.78 ± 0.78	4.04 ± 1.30	1.35 ± 1.42	
	Alternating	3.06 ± 0.39	2.99 ± 0.38	0.14 ± 0.16	0.15 ± 0.18
	PD-based	3.24 ± 0.38	3.16 ± 0.37	0.27 ± 0.20	
	h_a only	6.70 ± 1.47		-	

training, validation and test sets are identical to those of Section 3.3.1. Inputs are then scaled to be roughly in $[-1, 1]$. Here again, we use $f_k(\mathbf{x}_k) = \theta_0 \sin(\theta_1 x_0 x_1)$.

As in Section 3.3.1, Table 3 shows that hybrid models outperform their data-driven equivalents. PD-based methods usually yield more robust h_k estimations in the general unfiltered case, but struggle to line up with the alternate scheme in terms of predictive performance, except for GB-related models. For RF, this can be explained by a worse h_k estimation while for MLP we assume that it is due to h_a overfitting: in the alternate approach, it is optimized one epoch at a time, interleaved with one step on h_k, whereas that of PD-based methods is fully optimized (with identical complexities). Sequential and alternate approaches yield poor h_k estimations without filtering since A3 is not met, but the latter mitigates this w.r.t. the former, as was already observed

Table 4 Results for the *Overlapping* problem.

Method	$\hat{d}(h, y; TS)$ Unfiltered	Filtered	$\hat{d}(h, y; TS)$ Unfiltered	Filtered
	MLP		GB	
$f_k \to h_a$	0.35 ± 0.02	0.54 ± 0.04	0.51 ± 0.04	1.00 ± 0.12
Sequential	0.35 ± 0.01	0.59 ± 0.05	0.55 ± 0.07	1.07 ± 0.11
Alternate	0.35 ± 0.02	0.56 ± 0.05	0.54 ± 0.09	1.01 ± 0.11
PD-based	0.34 ± 0.02	0.56 ± 0.05	0.53 ± 0.05	0.99 ± 0.12
h_a only	0.37 ± 0.02		0.55 ± 0.07	

in Section 3.3.1. Input filtering degrades predictive performance for the sequential methods as they cannot counterbalance a poor h_k.

3.3.3 Overlapping additive structure (A2 and A3 not satisfied)

Let $y = \beta x_0^2 + \sin(\gamma x_0) + \delta x_1 + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 0.5^2), \beta = 0.2, \gamma = 1.5, \delta = 1$ and $\mathbf{x}$ sampled as in the *Correlated Linear* problem. We generate 50, 50 and 600 samples respectively for the training, validation and test sets. We define $f_k(\mathbf{x}_k) = \beta x_0^2$ and $f_a(\mathbf{x}) = \sin(\gamma x_0) + \delta x_1 + \varepsilon$. Hence, A2 and A3 do not hold. Even with $\hat{\beta} = \beta^*$, h_a still needs to compensate for $\sin(\gamma x_0)$. Therefore, filtering is expected to degrade performance for all hybrid approaches as $h_a(x_1)$ will never compensate for this gap, which we observe in Table 4, for both MLP and GB. Interestingly, sequential optimization seems to be the most impacted by filtering. It thus seems that having an interaction between the training of h_k and the training of h_a can still help to reduce the generalization error, even though $f_k(x_0)$ (and hence $h_k(x_0)$) is misspecified by design. Results for RF are not shown for the sake of space, but are similar to GB.

3.3.4 Impact of training sample size

In this experiment, we want to estimate the performance gap between hybrid methods and data-driven methods, for different sizes of the training sample. We measure the evolution of (1) and (2) for the different training algorithms, for increasing sample sizes of the *Correlated Friedman* problem. In particular, for a fixed training size, we generate 10 datasets using 10 different sets of values for $\theta_0, \ldots, \theta_5$. The same 10 validation and test samples are fixed across all training sample sizes, containing 300 and 600 samples each.

We observe in Figure 1 that the theoretically ideal $f_k \to h_a$ method generally performs better than any other training scheme, and significantly better for low-size regimes. Furthermore, hybrid models almost always outperform data-driven methods, except for PD-based training which struggles with very small sample sizes (30 training samples). This could be explained by the fact that, in this algorithm, the training of h_k highly depends on the quality of fit of h_a, which serves as surrogate model for the true function f. Nevertheless, with slightly more training samples, we notice that it performs as good as the other training schemes. As could be expected, all
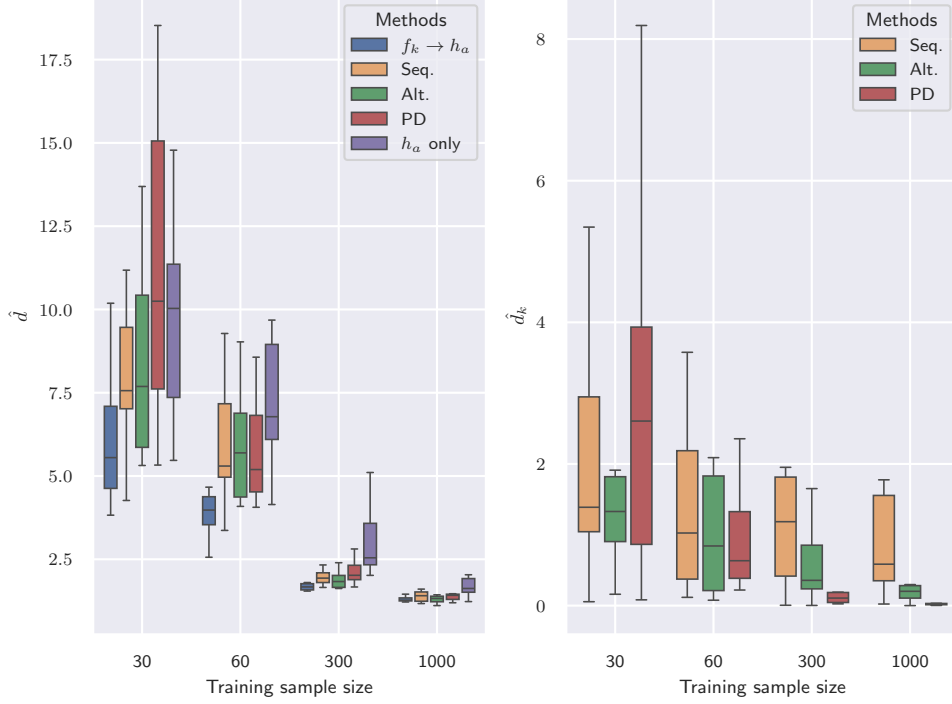
Fig. 1 Evolution of $\hat{d}$ (left) and $\hat{d}_k$ (right) w.r.t. the training sample size, on the *Correlated Friedman* problem. The boxplot summarizes the results over 10 test sets, using MLP as h_a.

training methods benefit from an increasing training sample size, and the gap between data-driven and hybrid models becomes smaller.

As observed for the generalization error, we also observe that the PD-based optimization yields worse estimates of h_k compared to the other hybrid approaches for very small sample sizes. However, it becomes competitive with the other methods at training samples of size 60, and eventually outperforms them at larger sample sizes.

Similar experiments have been conducted using RF and boosting with trees, and are reported in Appendix C.

3.3.5 Real regression problems

We now apply all methods on two real-world static datasets. As algebraic term h_k, we chose to use a linear prior on x_k, where x_k is the feature with the highest importance score in a RF model trained on the full dataset (assumed to be inaccessible at training time). As there is no guarantee that any of our assumptions are met (and in particular A1), we do not measure the distance d_k in (2) as it is deemed irrelevant.

We consider two settings for each dataset, inspired by Zhang et al. (2019). In the first setting (INT), the training and test sets are sampled from the same distribution $p(\mathbf{x}, y)$ whereas the second one (EXT) evaluates extrapolation performance for samples

Table 5 Variables used for the real-world datasets. For each dataset, the variable indicated in bold type is the one used in the linear prior (x_k).

Dataset	Name	Description
CCPP	**T**	**Ambient temperature [°C]**
	AP	Ambient pressure [mbar]
	RH	Relative humidity [-]
	V	Exhaust vacuum [cmHg]
CCS	Cement	Amount of cement in the mixture [kg/m^3]
	Blast Furnace Slag	Amount of blast furnace slag in the mixture [kg/m^3]
	Fly Ash	Amount of fly ash in the mixture [kg/m^3]
	Water	Amount of water in the mixture [kg/m^3]
	Superplasticizer	Amount of superplasticizer in the mixture [kg/m^3]
	Coarse Aggregate	Amount of coarse aggregate in the mixture [kg/m^3]
	Fine Aggregate	Amount of fine aggregate in the mixture [kg/m^3]
	Age	Day (1-365)
	Cement/Water	**Cement to water ratio [-]**

with unseen target values. If the linear prior is a reasonable assumption or, at the very least, the target increases or decreases monotonically with x_k, then we can expect hybrid methods to yield better results in the latter setting. In the INT setting for each dataset, we randomly select 100 samples for the learning set, 100 samples for the validation set and keep the rest as test set. For the EXT setting, we select the samples (one fourth of the dataset) with the lowest output values as test set. From the remaining samples, we randomly select 100 samples for the learning set and 100 samples for the validation set. For both INT and EXT settings, performance metrics are averaged over 10 different splits. We standardize both input and output variables.

The features for both datasets are described in Table 5. The *Combined Cycle Power Plant* (CCPP) dataset (Tfekci and Kaya, 2014) collects 9,568 measurements of net hourly electrical energy output for a combined cycle power plant, along with four hourly average input variables. The *Concrete Compressive Strength (CCS) dataset* (Yeh, 2007) is composed of 1,030 samples relating amounts of concrete components with the resulting compressive strength. As done in the work of Zhang et al. (2019), we introduce a new feature corresponding to the cement-to-water ratio.

From Table 6, it seems that introducing a linear prior does not yield any benefit in the interpolation setting, as all models perform equally well, which suggests that either the prior is not adequate or that A1 is not satisfied. In the extrapolation scenario, we can however observe that the linear prior allows to mitigate the impact of moving out of the distribution compared to data-driven models. Indeed, compared to the INT setting, the performance of all purely data-driven methods degrades in the EXT scenario, especially for GB and RF as their output predictions are bounded by the minimum target value observed in the training set. PD-based hybrid methods consistently outperform other hybrid approaches and are only slightly impacted, while sequential and alternating methods attain similar results.

13

Table 6 Results for the real-world datasets. We report the mean and standard deviation of $\hat{d}$ over the test sets.

		CCPP		CCS	
	Method	INT	EXT	INT	EXT
MLP	Sequential	0.07 ± 0.01	0.23 ± 0.13	0.25 ± 0.04	0.74 ± 0.24
	Alternating	0.07 ± 0.01	0.24 ± 0.10	0.24 ± 0.03	0.89 ± 0.36
	PD-based	0.07 ± 0.01	0.07 ± 0.01	0.25 ± 0.03	0.38 ± 0.04
	h_a only	0.07 ± 0.01	0.36 ± 0.21	0.25 ± 0.05	0.83 ± 0.30
GB	Sequential	0.08 ± 0.01	0.35 ± 0.07	0.21 ± 0.02	0.91 ± 0.24
	Alternating	0.08 ± 0.01	0.41 ± 0.17	0.20 ± 0.03	0.85 ± 0.28
	PD-based	0.08 ± 0.02	0.10 ± 0.01	0.22 ± 0.02	0.31 ± 0.06
	h_a only	0.10 ± 0.01	0.95 ± 0.18	0.25 ± 0.02	1.54 ± 0.28
RF	Sequential	0.07 ± 0.01	0.28 ± 0.07	0.20 ± 0.02	1.06 ± 0.31
	Alternating	0.07 ± 0.01	0.32 ± 0.12	0.20 ± 0.02	1.00 ± 0.32
	PD-based	0.07 ± 0.01	0.08 ± 0.01	0.20 ± 0.02	0.30 ± 0.08
	h_a only	0.08 ± 0.01	0.99 ± 0.19	0.25 ± 0.02	1.89 ± 0.39

4 Dynamical systems forecasting

In this section, we focus on the problem of dynamical systems forecasting. We start by defining more formally the problem in Section 4.1. In Section 4.2, we explain how PD-based training can be extended to the dynamical system setting and we present the hybrid optimization processes that it will eventually be compared to on several physical systems in Section 4.3.

4.1 Problem statement

In this section, we target the problem of dynamical systems forecasting, i.e. predicting the evolution of systems whose dynamics are driven by:

$$\dot{\mathbf{x}}_t := \frac{d\mathbf{x}_t}{dt} = f(\mathbf{x}_t), \tag{10}$$

where the state $\mathbf{x}_t$ can be vector-valued, i.e. $\mathbf{x}_t \in \mathbb{R}^d$, or a d-dimensional vector field defined over some spatial domain, and $f(\mathbf{x}_t)$ has the same dimension as $\mathbf{x}_t$. We assume access to a finite set LS of N observed state trajectories over a given time interval $[0, T]$, with T the time horizon, i.e. $LS = \{\mathbf{x}_0^{(i)}, \ldots, \mathbf{x}_T^{(i)}\}_{i=1}^N$. For clarity of notation, we use a single T although the time horizon can vary according to the considered system and between the N trajectories. Specifically, we consider systems whose dynamics f can be additively decomposed so as to satisfy A1, i.e.:

$$f(\mathbf{x}_t) = f_k(\mathbf{x}_{k,t}) + f_a(\mathbf{x}_t). \tag{11}$$

As in Section 3.1, we assume partial knowledge through some parametric term $h_k^{\theta_k} \in \mathcal{H}_k$ with trainable parameters θ_k, such that for the optimal θ_k^* we have $h_k^{\theta_k^*} = f_k$. The

residual function f_a is modeled by some ML component $h_a^{\theta_a}$, such that the final model is given by $h = h_k + h_a$. Similarly to the static setting, we introduce a constant term γ to the physical term $h_k^{\theta_k}$, such that we fit $h_k^{\theta_k} + \gamma$.

We seek to minimize the following two distances:

$$d(h, f) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} ||h(\mathbf{x}) - f(\mathbf{x})||^2, \tag{12}$$

$$d_k(h_k^{\theta_k}, f_k) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \{(h_k^{\theta_k}(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2\}, \tag{13}$$

where $p(\mathbf{x})$ is the distribution over states. The first one measures the distance between the true dynamics f and its approximation h. The second distance d_k measures how well the tuned h_k approximates f_k. In practice, minimizing (12) translates into minimizing the distance between observed trajectories and trajectories obtained by integration of h:

$$\hat{d}(h, f; LS) = \frac{1}{N} \frac{1}{T} \sum_{i=1}^{N} \sum_{t=1}^{T} (\hat{\mathbf{x}}_t^{(i)} - \mathbf{x}_t^{(i)})^2, \tag{14}$$

where $\hat{\mathbf{x}}_t^{(i)}$ is the solution of the integration of h with initial condition $x_0^{(i)}$ up to time t. In our experiments, we measure (13) on some test set TS as:

$$\hat{d}(h_k, f_k; TS) = \frac{1}{N} \frac{1}{T} \sum_{i=1}^{N} \sum_{t=1}^{T} (h_k(\mathbf{x}_{k,t}^{(i)}) - f_k(\mathbf{x}_{k,t}^{(i)}))^2. \tag{15}$$

4.2 Training methods

Extending the algorithms of Section 3.2 to dynamical problems requires the integration of h to obtain the solutions $\hat{\mathbf{x}}_t^{(i)}$, which we do through the Neural ODE approach (Chen et al., 2018). Contrary to the static experiments, we solely relied on ML models that allow backpropagation through time (allowing for easy integration with Neural ODE), thereby excluding tree-based models. For that reason, we can now introduce another hybrid training scheme in the comparisons, namely joint optimization of h_k and h_a, that was proposed by Yin et al. (2021). Note that this optimization algorithm could not be used in the static experiments due to the fact that tree-based methods cannot be trained jointly with the parametric component h_k.

Given the model approximation error observed on h_k for the sequential scheme in previous results (see Table 2 and Table 3), and assuming that most dynamical systems exhibit correlation between f_k and f_a through time, we choose to omit the sequential optimization scheme and find filtering irrelevant in such a configuration. Hence, we compare our PD-based hybrid scheme with the joint and alternate optimization of h_k and h_a, as defined by Yin et al. (2021) and Donà et al. (2022) respectively. As for the static case, we compare all schemes to the theoretically best setting of fitting h_a on $f - f_k$, i.e., assuming that the algebraic term if perfectly known.

4.2.1 Joint optimization

In their work, Yin et al. (2021) optimize $h_k^{\theta_k}$ and $h_a^{\theta_a}$ jointly to minimize Equation (12). They introduce a regularizing criterion on h_a such that its contribution to the total

model $h_k + h_a$ is kept minimal, ensuring existence and uniqueness of the minimizing pair (h_k, h_a). We describe the non-regularized joint optimization process in Algorithm 4 (Appendix E). More details about the regularizer can be found in (Yin et al., 2021).

4.2.2 Alternate optimization

As described in Section 3.2.2, $(\hat{\theta}_k, \hat{\gamma})$ is initialized as the minimizer of $d(h_k^{\theta_k}, f)$, then this optimization scheme extends the previous one by alternating training steps between h_k and h_a. Donà et al. (2022) introduce an additional constraint on h_k to keep it close to its initialization in early training. The non-regularized training scheme is described in Algorithm 6 (Appendix E). Details about the regularizer can be found in (Donà et al., 2022).

4.2.3 Partial-dependence-based optimization

Our optimization scheme can be easily extended to the dynamical setting (see Algorithm 2). We first fit any ML model $h_a^{\theta_a}$ on observed trajectories with Neural ODE, which we denote $\mathrm{fit}_{\mathrm{NODE}}^{h_a}$. At this stage, $h_a^{\hat{\theta}_a}$ models the complete system dynamics f, hence it can be evaluated for multiple $\mathbf{x}_{k,t}$ values to obtain an initial approximation of $f_k(\mathbf{x}_{k,t})$ through $\widehat{PD}(h_a, \mathbf{x}_{k,t}; LS)$, on which we can fit $h_k^{\theta_k} + \gamma$ as for static regression problems, yielding an initial approximation $h_k^{\hat{\theta}_k} + \hat{\gamma}$. Due to potential residual dependencies, we then alternate between several epochs of fitting $h_a^{\theta_a}$ with Neural ODE to minimize $d(h_a^{\theta_a} + h_k^{\hat{\theta}_k} + \hat{\gamma}, f)$ (keeping $(\hat{\theta}_k, \hat{\gamma})$ fixed) and several epochs of fitting $h_k^{\theta_k} + \gamma$ on $\widehat{PD}(h_a^{\hat{\theta}_a}, \mathbf{x}_{k,t}; LS) + h_k^{\hat{\theta}_k} + \hat{\gamma}$, with $\hat{\theta}_k$, $\hat{\gamma}$ and $\hat{\theta}_a$ the current parameter vectors.

Algorithm 2 Partial Dependence Optimization - Dynamical problems

---

$\quad$ **Input:** $LS = \{(\mathbf{x}_0^{(i)}, \ldots, \mathbf{x}_T^{(i)})\}_{i=1}^N$

$\quad \hat{\theta}_a^{(0)} \leftarrow \mathrm{fit}_{\mathrm{NODE}}^{h_a}(LS; h_k = 0)$

$\quad (\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}) \leftarrow \mathrm{fit}^{h_k + \gamma}(\{(\mathbf{x}_{k,t}^{(i)}, \widehat{PD}(h_a^{\hat{\theta}_a^{(0)}}, \mathbf{x}_{k,t}^{(i)}; LS))\}_{i,t=1}^{N,T})$

$\quad$ **for** $n = 1$ **to** N_r **do**

$\qquad \hat{\theta}_a^{(n)} \leftarrow \mathrm{fit}_{\mathrm{NODE}}^{h_a}(LS; h_k = h_k^{\hat{\theta}_k^{(n-1)}} + \hat{\gamma}^{(n-1)})$

$\qquad (\hat{\theta}_k^{(n)}, \hat{\gamma}^{(n)}) \leftarrow \mathrm{fit}^{h_k + \gamma}(\{(\mathbf{x}_{k,t}^{(i)}, h_k^{\hat{\theta}_k^{(n-1)}}(\mathbf{x}_{k,t}^{(i)}) + \hat{\gamma}^{(n-1)} + \widehat{PD}(h_a^{\hat{\theta}_a^{(n)}}, \mathbf{x}_{k,t}^{(i)}; LS))\}_{i,t=1}^{N,T})$

$\quad$ **end for**

$\quad \hat{\theta}_a^{(N_r+1)} \leftarrow \mathrm{fit}_{\mathrm{NODE}}^{h_a}(LS; h_k = h_k^{\hat{\theta}_k^{(N_r)}} + \hat{\gamma}^{(N_r)})$

$\quad$ **return** $h_k^{\hat{\theta}_k^{(N_r)}} + \hat{\gamma}^{(N_r)} + h_a^{\hat{\theta}_a^{(N_r+1)}}$

4.3 Experiments

We compare the different training methods on three simulated dynamical systems that were used for evaluating related methods (Yin et al., 2021; Donà et al., 2022). We

measure performance through log-estimates of (14) and estimates of (15) on a test set TS, which we respectively denote log-$\hat{d}(h, f; TS)$ and $\hat{d}_k(h_k^{\theta_k}, f_k; TS)$ (lower is better). For hybrid approaches, we use as h_a either a multilayer perceptron (MLP) or a convolutional neural network (CNN), depending on the system. We compare these hybrid models to a standard data-driven framework that uses only h_a. Architectures (e.g. for MLP, the number of layers and neurons) are optimized for each training method (unless clearly specified) on the validation set. Details about the hyperparameter optimization procedure and their optimal values are given in Appendix B. We monitor (14) on a validation set and select the model that reaches the lowest validation loss. Datasets and optimization details are provided in Appendix D. For conciseness, we only report results obtained using the optimization scheme as exactly described in each reference work and compare the impact of using the specific regularization criterion or not (if relevant). Extended results for additional configurations of the methods are provided in Appendix D.4.

4.3.1 *Lotka-Volterra* system

Lotka-Volterra equations model the evolution of preys x and predators y according to:

$$\begin{cases} \dot{x} &= \alpha x - \beta xy \\ \dot{y} &= -\delta y + \gamma xy \end{cases} \iff \begin{cases} \dot{p} &= \alpha - \beta e^q \\ \dot{q} &= -\delta + \gamma e^p \end{cases}$$

where $p = \ln(x)$ and $q = \ln(y)$. Due to training stability reasons, we did not manage to fit any method on the left-hand side system. Therefore, we choose to focus on the right-hand side system, where $\mathbf{x} = (p, q)$ is the system state. We assume $h_k^{\theta_k}(p, q) = (-\theta_k^1 e^q, 0)$ and $h_a^{\theta_a} = (\alpha, -\delta + \gamma e^p)$.

Table 7 Results for the *Lotka-Volterra* system, averaged over 5 different initializations of h. Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Init. (resp. No Init.) refers to initializing h_k as the minimizer of $d(h_k, f)$ (resp. randomly).

Method	log-$\hat{d}(h, f; TS)$		$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$	
	No Reg.	Reg.	No Reg.	Reg.
$f_k \rightarrow h_a$	-5.45 ± 0.02		-	
Joint (No Init.)	-5.22 ± 0.04	-5.45 ± 0.06	$2.51{\times}10^{-1} \pm 3.90{\times}10^{-2}$	$1.17{\times}10^{-6} \pm 1.47{\times}10^{-13}$
Alternate (Init.)	-5.19 ± 0.04	$\mathbf{-5.52 \pm 0.05}$	$3.79{\times}10^{-1} \pm 4.49{\times}10^{-2}$	$1.35{\times}10^{-1} \pm 7.19{\times}10^{-3}$
PD-based (No Init.)	-5.41 ± 0.04		$\mathbf{1.05{\times}10^{-7} \pm 9.51{\times}10^{-15}}$	
h_a only	-5.02 ± 0.09		-	

In terms of trajectory estimation, we can observe from Table 7 that all hybrid optimization schemes perform very similarly, although alternate training performs slightly better. We can also note that all training methods yield better estimators than the purely data-driven setting. As concerns the fitting of h_k, results suggest that our PD-based training procedure produces a more robust and accurate estimation of f_k.

We can also study the effect of the regularizers introduced by Yin et al. (2021) and Donà et al. (2022). For this specific problem, joint and alternate optimization

combined with their regularizers seem to improve both the estimation of f_k and the final predictive performance.

4.3.2 *Damped Pendulum* system

The evolution of the angle $\theta(t)$ of a damped pendulum system is described as follows:

$$\frac{d^2\theta}{dt^2} + \omega_0^2 \sin\theta + \xi\frac{d\theta}{dt} = 0, \quad \Longleftrightarrow \quad \begin{cases} \dot{\theta} &= \frac{d\theta}{dt} \\ \ddot{\theta} &= -\omega_0^2\sin\theta - \xi\frac{d\theta}{dt} \end{cases}$$

where $\mathbf{x} = (\theta, \frac{d\theta}{dt})$ is the system state. We assume $h_k^{\theta_k}(\theta, \frac{d\theta}{dt}) = (\frac{d\theta}{dt}, -\theta_k\sin\theta)$, and $h_a^{\theta_a}$ accounts for the damping term $-\xi d\theta/dt$.

Table 8 Results for the *Damped Pendulum* system, averaged over 5 different initializations of h. Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Init. (resp. No Init.) refers to initializing h_k as the minimizer of $d(h_k, f)$ (resp. randomly).

Method	log-$\hat{d}(h, f; TS)$		$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$	
	No Reg.	Reg.	No Reg.	Reg.
$f_k \to h_a$	$-\mathbf{8.75 \pm 0.01}$		-	
Joint (No Init.)	-7.68 ± 0.03	-5.98 ± 0.04	$2.29{\times}10^{-1} \pm 4.47{\times}10^{-3}$	$5.51{\times}10^{-3} \pm 2.36{\times}10^{-6}$
Alternate (Init.)	-7.36 ± 0.02	-6.91 ± 0.00	$1.61{\times}10^{-1} \pm 4.09{\times}10^{-3}$	$5.90{\times}10^{-3} \pm 2.64{\times}10^{-6}$
PD-based (No Init.)	-8.28 ± 0.04		$\mathbf{2.42{\times}10^{-5} \pm 4.57{\times}10^{-12}}$	
h_a only	-7.28 ± 0.01		-	

Table 8 highlights that our PD-based optimization predicts more accurate trajectories than the other hybrid training methods. Similarly, all training methods yield better estimators than the purely data-driven setting, provided we do not regularize. For this system, we see that the baseline $f_k \to h_a$ is now the optimal training approach. Concerning the estimation of h_k, we observe that our hybrid training scheme yields the best estimations. Contrary to the previous system, regularizing joint and alternate optimization improves the resulting h_k, but now degrades final predictive performance. This underlines the fact that there exists an interesting trade-off to investigate for both methods, but also that they are very sensitive to the chosen regularization hyperparameter values.

4.3.3 *Reaction-Diffusion* system

This 2-dimensional system is driven by the following PDEs (Klaasen and Troy, 1984):

$$\begin{cases} \frac{\delta u}{\delta t} &= \alpha\Delta u + R_u(u, v; \gamma) \\ \frac{\delta v}{\delta t} &= \beta\Delta v + R_v(u, v) \end{cases}$$

where $\mathbf{x} = (u, v)$ is the system state defined over some rectangular domain with circular boundary conditions. Δ is the Laplace operator and α, β are the diffusion coefficients for both quantities. $R_u(u, v; \gamma) = u - u^3 - \gamma - v$ and $R_v(u, v) = u - v$ are the local reaction terms. We assume $h_k^{\theta_k}(u, v) = (\theta_k^1\Delta u, \theta_k^2\Delta v)$, and $h_a^{\theta_a}$ accounts for the

reaction terms R_u and R_v. Contrary to the two previous experiments, due to time and computational constraints, we did not optimize hyperparameters. Instead, all training schemes were given the same set of hyperparameters that we deemed reasonable.

Table 9 Results for the *Reaction-Diffusion* system, averaged over 5 different initializations of h. Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Init. (resp. No Init.) refers to initializing h_k as the minimizer of $d(h_k, f)$ (resp. randomly).

| Method | log-$\hat{d}(h, f; TS)$ | | $\hat{d}_k(h_k^{\theta_k}, f_k; TS)$ | |
	No Reg.	Reg.	No Reg.	Reg.
$f_k \to h_a$	-3.43 ± 0.11		-	
Joint (No Init.)	$\mathbf{-3.62 \pm 0.02}$	-2.96 ± 0.01	$\mathbf{1.17{\times}10^{-5} \pm 1.42{\times}10^{-10}}$	$5.45{\times}10^{-4} \pm 2.64{\times}10^{-7}$
Alternate (Init.)	-2.99 ± 0.02	-2.97 ± 0.01	$4.80{\times}10^{-5} \pm 2.81{\times}10^{-9}$	$1.89{\times}10^{-3} \pm 1.67{\times}10^{-6}$
PD-based (No Init.)	-3.09 ± 0.02		$1.64{\times}10^{-3} \pm 1.01{\times}10^{-7}$	
h_a only	-2.86 ± 0.01		-	

From Table 9, we observe that the best performing scheme is the non-regularized joint training of h_k and h_a, as opposed to the previous problems. As before, introducing the regularizers defined in both reference works induces a drop in final performance, still more controlled for alternate optimization. This drop is also observed in the estimation of f_k (which seems optimal for the non-regularized configuration), confirming that such regularizing criteria are very sensitive to optimize.

PD-based training yields reasonable trajectory estimates, but the quality of its estimation of h_k is no longer the best amongst all. Indeed, contrary to the two previous systems, it is clear that A2 is no longer satisfied as $f_a(u, v) = (u - u^3 - \gamma - v, u - v)$ includes variables present in f_k. The latter setting represents a limitation of the PD-based optimization scheme as its estimation of the physical component is now effectively impacted, compared to the other hybrid training methods.

5 Conclusion

We study several methods for training hybrid additive models on static supervised regression problems, as well as on dynamical systems. On static problems, we empirically show that trends observed for neural networks also apply for the non-parametric tree-based approaches, in terms of predictive performance as well as in the estimation of the algebraic known function. We introduce claims related to the convergence of these hybrid approaches and verify their soundness on illustrative experiments. In particular, for such static additive problems where both terms are functions of disjoint and independent input features, we hypothesize that sequential and alternate training asymptotically converge to the true parametric component.

We present a new hybrid approach leveraging partial dependence and show its competitiveness against sequential and alternate optimization schemes on both synthetic and real-world static problems. We extend the modeling framework to dynamical systems forecasting and show that PD-based training also yields robust and accurate models in such setting. We highlight its benefits in estimating the parametric prior and show that it alleviates both the risk of the ML term to dominate the known term

and the need for assuming independent input features sets, although the quality of its parametric estimator may by degraded when there is a clear overlap of input features between f_k and f_a. Another advantage of the PD-based training scheme is the reduced need for hyperparameter tuning compared to the alternate approach, which introduces multiple regularization coefficients that must be jointly optimized.

As a more general conclusion, hybrid methods are shown in our experiments to improve predictive performance with respect to ML-only models, although not always very significantly. The main benefit of the alternate and PD-based methods over the simple sequential approach is that they provide better estimators of the algebraic term, especially when filtering is relevant and can be applied. As concerns dynamical systems, such hybrid methods yield more interpretable and accurate models than purely data-driven models.

As future work, we are interested in studying theoretical conditions of optimality for our PD-based hybrid training and other approaches, in the context of dynamical systems. We also plan to try and extend current hybrid methods tackling dynamical systems forecasting so that they can be used with tree-based methods.

Declarations

Competing interests

The authors declare they have no financial nor non-financial interests.

References

Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)

Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)

Claes, Y., Huynh-Thu, V.A., Geurts, P.: Knowledge-guided additive modeling for supervised regression. In: Discovery Science - 26th International Conference, DS 2023, Porto, Portugal, October 9-11, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14276, pp. 64–78. Springer, Berlin (2023)

Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. Advances in Neural Information Processing Systems (2018)

Chen, J., Yu, K., Hsing, A., Therneau, T.M.: A partially linear tree-based regression model for assessing complex joint gene–gene and gene–environment effects. Genetic

Epidemiology: The Official Publication of the International Genetic Epidemiology Society **31**(3), 238–251 (2007)

De Bézenac, E., Pajot, A., Gallinari, P.: Deep learning for physical processes: Incorporating prior scientific knowledge. Journal of Statistical Mechanics: Theory and Experiment **2019**(12), 124009 (2019)

Donà, J., Déchelle, M., Levy, M., Gallinari, P.: Constrained physical-statistics models for dynamical system identification and prediction. In: ICLR 2022-The Tenth International Conference on Learning Representations (2022)

Daw, A., Karpatne, A., Watkins, W.D., Read, J.S., Kumar, V.: Physics-guided neural networks (pgnn): An application in lake temperature modeling. In: Knowledge Guided Machine Learning, pp. 353–372. Chapman and Hall/CRC, New York (2022)

Friedman, J.H., Grosse, E., Stuetzle, W.: Multidimensional additive spline approximation. SIAM Journal on Scientific and Statistical Computing **4**(2), 291–301 (1983)

Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189–1232 (2001)

Friedman, J.H.: Stochastic gradient boosting. Computational statistics & data analysis **38**(4), 367–378 (2002)

Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? Advances in Neural Information Processing Systems **35**, 507–520 (2022)

Greene, W.H.: Econometric Analysis. Econometric Analysis. Pearson, Upper Saddle River, NJ (2018)

Hu, G., Mao, Z., He, D., Yang, F.: Hybrid modeling for the prediction of leaching rate in leaching process based on negative correlation learning bagging ensemble algorithm. Computers & chemical engineering **35**(12), 2611–2617 (2011)

Johansen, T.A., Foss, B.A.: Representing and learning unmodeled dynamics with neural network memories. In: IEEE American Control Conf., pp. 3037–3043 (1992)

Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

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

Klaasen, G.A., Troy, W.C.: Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. SIAM Journal on Applied Mathematics **44**(1), 96–110 (1984)

Kramer, M.A., Thompson, M.L., Bhagat, P.M.: Embedding theoretical models in neural networks. In: 1992 American Control Conf., pp. 475–479 (1992). IEEE

Mehta, V., Char, I., Neiswanger, W., Chung, Y., Nelson, A., Boyer, M., Kolemen, E., Schneider, J.: Neural dynamical systems: Balancing structure and flexibility in physical prediction. In: 2021 60th IEEE Conference on Decision and Control (CDC), pp. 3735–3742 (2021). IEEE

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

Qian, Z., Zame, W., Fleuren, L., Elbers, P., Schaar, M.: Integrating expert odes into neural odes: Pharmacology and disease progression. Advances in Neural Information Processing Systems **34**, 11364–11383 (2021)

Thompson, M.L., Kramer, M.A.: Modeling chemical processes using prior knowledge and neural networks. AIChE Journal **40**(8), 1328–1340 (1994)

Tfekci, P., Kaya, H.: Combined Cycle Power Plant. UCI Machine Learning Repository (2014)

Takeishi, N., Kalousis, A.: Physics-integrated variational autoencoders for robust and interpretable generative modeling. Advances in Neural Information Processing Systems **34**, 14809–14821 (2021)

Von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., *et al.*: Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Transactions on Knowledge and Data Engineering **35**(1), 614–633 (2021)

Wehenkel, A., Behrmann, J., Hsu, H., Sapiro, G., Louppe, G., Jacobsen, J.-H.: Improving generalization with physical equations. Machine Learning and the Physical Sciences: Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS) (2022)

Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V.: Integrating scientific knowledge with machine learning for engineering and environmental systems. ACM Computing Surveys **55**(4), 1–37 (2022)

Wouwer, A.V., Renotte, C., Bogaerts, P.: Biological reaction modeling using radial basis function networks. Computers & chemical engineering **28**(11), 2157–2164

(2004)

Yin, Y., Ayed, I., Bézenac, E., Baskiotis, N., Gallinari, P.: Leads: Learning dynamical systems that generalize across environments. Advances in Neural Information Processing Systems **34**, 7561–7573 (2021)

Yeh, I.-C.: Concrete Compressive Strength. UCI Machine Learning Repository (2007)

Yin, Y., Le Guen, V., Dona, J., Bézenac, E., Ayed, I., Thome, N., Gallinari, P.: Augmenting physical models with deep networks for complex dynamics forecasting. Journal of Statistical Mechanics: Theory and Experiment **2021**(12), 124012 (2021)

Yu, K., Wheeler, W., Li, Q., Bergen, A.W., Caporaso, N., Chatterjee, N., Chen, J.: A partially linear tree-based regression model for multivariate outcomes. Biometrics **66**(1), 89–96 (2010)

Zhang, H., Nettleton, D., Zhu, Z.: Regression-enhanced random forests. arXiv preprint arXiv:1904.10416 (2019)

Appendix A   Optimal model under A2 and A3

Let us recall the regression problem, where $y \in \mathbb{R}$ can be decomposed into the addition of two independent terms:

$$y = f_k(\mathbf{x}_k) + f_a^r(\mathbf{x}_a) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2), \quad \mathbf{x}_k \cup \mathbf{x}_a = \mathbf{x}, \mathbf{x}_k \cap \mathbf{x}_a = \emptyset, \mathbf{x}_k \perp\!\!\!\perp \mathbf{x}_a.$$

For clarity, let us denote by $\mathbb{E}_{\mathbf{x}}$ the subsequent expectations over the input space $\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \{\cdot\}$. We have:

$$\hat{f}_k^* = \arg \min_{\hat{f}_k \in \hat{\mathcal{F}}_k} d(\hat{f}_k, y) = \arg \min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathbb{E}_{\mathbf{x}, \varepsilon} \left\{ \left(\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k) - f_a^r(\mathbf{x}_a) - \varepsilon \right)^2 \right\}$$

$$= \arg \min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathbb{E}_{\mathbf{x}_k} \left\{ (\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2 \right\} + \mathbb{E}_{\mathbf{x}_a, \varepsilon} \left\{ (f_a^r(\mathbf{x}_a) + \varepsilon)^2 \right\}$$

$$- \mathbb{E}_{\mathbf{x}, \varepsilon} \left\{ 2(\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k))(f_a^r(\mathbf{x}_a) + \varepsilon) \right\}.$$

The second term is independent w.r.t. $\hat{f}_k$ and thus has no impact on the minimization. Moreover, since $\mathbf{x}_k \perp\!\!\!\perp \mathbf{x}_a$, the last term writes as the product of two expectations, one of which is constant w.r.t. $\hat{f}_k$. We thus have:

$$\hat{f}_k^* = \arg \min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathbb{E}_{\mathbf{x}_k} \left\{ (\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2 \right\} - 2C \mathbb{E}_{\mathbf{x}_k} \left\{ (\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k)) \right\}, \quad \text{(A1)}$$

with $C = \mathbb{E}_{\mathbf{x}_a, \varepsilon} \{(f_a^r(\mathbf{x}_a) + \varepsilon)\} = \mathbb{E}_{\mathbf{x}_a} \{f_a^r(\mathbf{x}_a)\}$. Cancelling the derivative of (A1) w.r.t. $\hat{f}_k$, we obtain the optimal model $\hat{f}_k^*(\mathbf{x}_k) = f_k(\mathbf{x}_k) + C$, for every $\mathbf{x}_k \in \mathcal{X}_k$.

Appendix B   Model architectures

We list below the hyperparameter values that have been used in all experiments.

B.1   Static regression problems

Model hyperparameters for the static regression problems are reported in Table B1. These values have been chosen on the basis of preliminary small-scale experiments, with the main aim of obtaining a good tradeoff between predictive performance and computation times of the full experiments. Obviously better performance can be obtained for each of the methods with a more intensive hyper-parameter tuning. We used PyTorch (Paszke et al., 2019) for MLP, scikit-learn (Pedregosa et al., 2011) for RF and xgboost (Chen and Guestrin, 2016) for GB. Unspecified parameters keep their default values. Learning rates for training h_k and MLP are set to 0.005. H is the number of hidden layers in MLP and W the number of neurons per hidden layer. T is the number of trees in GB and RF, d the maximum tree depth and mss the minimum number of samples required to split an internal tree node.

Table B1 Model hyperparameters, for each experiment.

Problem	MLP	GB	RF
(Correlated) Friedman problems	$H = 2, W = 15$	$T = 700, d = 2$	$T = 500, mss = 5$
Linear & *Overlapping* problems	$H = 2, W = 10$	$T = 400, d = 2$	$T = 500, mss = 5$
Real-world problems	$H = 2, W = 30$	$T = 300, d = 2$	$T = 200, mss = 5$

Table B2 Hyperparameter values tested for the *Lotka-Volterra* and *Damped Pendulum* systems.

Hyperparameter			
Learning Rate (LR)	5×10^{-4}	1×10^{-4}	1×10^{-5}
Num. hidden layers	1	2	-
Hidden size	16	32	64
λ_h - Joint	1	10	20
λ_h - Alternate	1×10^{-1}	1×10^{-2}	1×10^{-3}
λ_{h_k} - Alternate	1×10^{-1}	1×10^{-2}	1×10^{-3}
λ_{h_a} - Alternate	1×10^{-1}	1×10^{-2}	1×10^{-3}

B.2   Dynamical systems forecasting

Model hyperparameters for the dynamical forecasting problems are reported in Tables B3, B4 and B5. We used `PyTorch` for MLP and CNN. Tested hyperparameters are reported in Table B2. Given that we only optimized hyperparameters for the *Lotka-Volterra* and *Damped Pendulum* systems, architectural hyperparameters solely refer to those of MLPs. According to the algorithm described in (Yin et al., 2021) and (Donà et al., 2022), λ_h is optimized for joint optimization whereas λ_h, λ_{h_k} and λ_{h_a} are optimized for alternate training.

For each problem, for a given optimization scheme and for each of the five different initializations of h, we tested every combination of hyperparameters and evaluated it on the corresponding validation set. As optimal hyperparameters may vary from one seed to the other, we checked all five optimal configurations of the given training method and reported the most frequent one. When no clear majority could be established, we reported several optimal values for the same hyperparameter. Note that the optimal hyperparameters across the five seeds were most of the time identical.

Table B3 Optimal model hyperparameters (MLP), for the *Lotka-Volterra* system.

Method	LR	Num. layers	Hidden size	λ_h	λ_{h_k}	λ_{h_a}
$f_k \rightarrow h_a$	5×10^{-4}	2	64	-	-	-
Joint - No Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Joint - No Init. & Reg.	1×10^{-5}	2	64	10/20	-	-
Joint - Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Joint - Init. & Reg.	1×10^{-5}	2	64	20	-	-
Alternate - No Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Alternate - No Init. & Reg.	5×10^{-4}	2	64	0.1	0.001	0.001
Alternate - Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Alternate - Init. & Reg.	5×10^{-4}	2	64	0.1	0.01	0.001
PD-based - No Init.	5×10^{-4}	2	64	-	-	-
PD-based - Init.	5×10^{-4}	2	64	-	-	-
h_a only	5×10^{-4}	2	64	-	-	-

Table B4 Optimal model hyperparameters (MLP), for the *Damped Pendulum* system.

| Method	LR	Num. layers	Hidden size	λ_h	λ_{h_k}	λ_{h_a}
$f_k \to h_a$	5×10^{-4}	2	64	-	-	-
Joint - No Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Joint - No Init. & Reg.	1×10^{-4}	1/2	32/16	20	-	-
Joint - Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Joint - Init. & Reg.	1×10^{-4}	1/2	16	20	-	-
Alternate - No Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Alternate - No Init. & Reg.	5×10^{-4}	2	64	0.1	0.001	0.001
Alternate - Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Alternate - Init. & Reg.	5×10^{-4}	2	64	0.1	0.001	0.001
PD-based - No Init.	5×10^{-4}	2	64	-	-	-
PD-based - Init.	5×10^{-4}	2	64	-	-	-
h_a only	5×10^{-4}	2	64	-	-	-

Table B5 Model hyperparameters (CNN), for the *Reaction-Diffusion* system.

Method	LR	Num. layers	Hidden channels	λ_h	λ_{h_k}	λ_{h_a}
$f_k \to h_a$	1×10^{-4}	3	8	-	-	-
Joint - No Init. & No Reg.	1×10^{-4}	3	8	-	-	-
Joint - No Init. & Reg.	1×10^{-4}	3	8	20	-	-
Joint - Init. & No Reg.	1×10^{-4}	3	8	-	-	-
Joint - Init. & Reg.	1×10^{-4}	3	8	20	-	-
Alternate - No Init. & No Reg.	1×10^{-4}	3	8	-	-	-
Alternate - No Init. & Reg.	1×10^{-4}	3	8	0.1	0.01	0.001
Alternate - Init. & No Reg.	1×10^{-4}	3	8	-	-	-
Alternate - Init. & Reg.	1×10^{-4}	3	8	0.1	0.01	0.001
PD-based - No Init.	1×10^{-4}	3	8	-	-	-
PD-based - Init.	1×10^{-4}	3	8	-	-	-
h_a only	1×10^{-4}	3	8	-	-	-

Appendix C Impact of training sample size - Additional results

Similarly to MLP, we observe in Figure C1 and Figure C2 that hybrid methods with boosting outperform data-driven methods. Once again, PD-based training seems to struggle with very low sample sizes, although the gap with other hybrid schemes is more moderate and the performance of boosting alone is worse than that of MLP alone. Whatever the training sample size, the performance of hybrid methods with boosting is on par (and sometimes better) with that of MLP, both for generalization error and estimates of h_k.

The same observations can be made for RF, even though its performance is slightly worse than its MLP or boosting counterparts, whatever the training sample size. Still, it is interesting to note that hybrid methods using RF yield the best estimates of h_k for the smallest sample size.
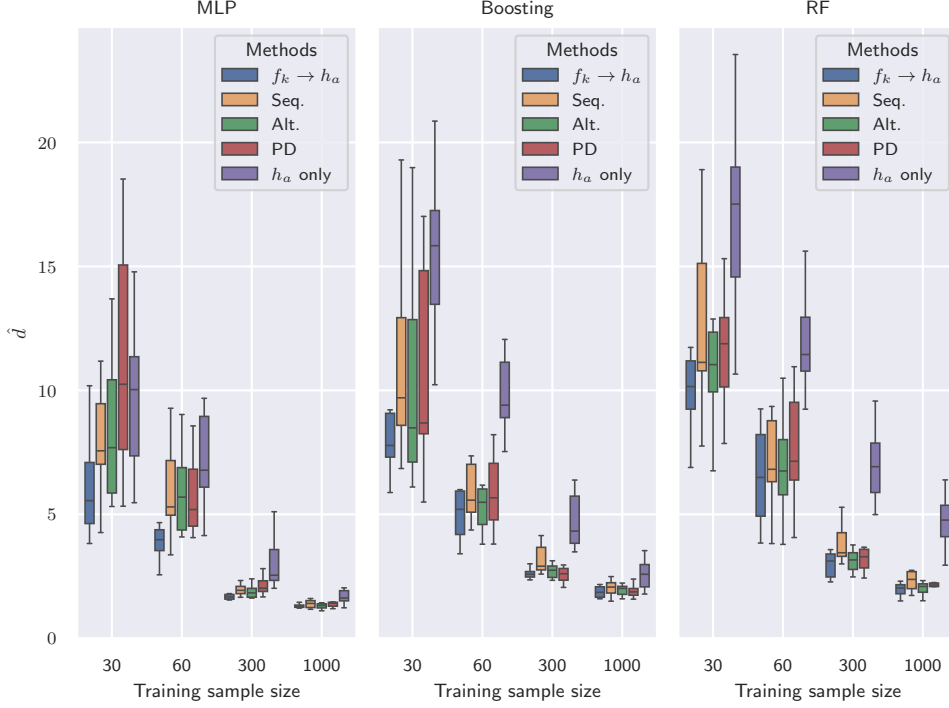
Fig. C1 Evolution of $\hat{d}$ w.r.t. the training sample size, on the *Correlated Friedman* problem. The boxplot summarizes the results over 10 test sets.

Appendix D   Dynamical experiments details

D.1   *Lotka-Volterra* system

The system is driven by the following dynamics:

$$\begin{cases} \dot{x} & = \alpha x - \beta xy \\ \dot{y} & = -\delta y + \gamma xy \end{cases}$$

where prey and predator quantities x and y can be transformed as $p = \ln(x), q = \ln(y)$ such that

$$\begin{cases} \dot{p} & = \alpha - \beta e^q \\ \dot{q} & = -\delta + \gamma e^p \end{cases}$$

where $\mathbf{x} = (p, q)$ is the system state.

We simulate a dataset of 200 trajectories spanning 401 timesteps (including the initial state $\mathbf{x}_0$), over a time interval of $[0, 20]$s. We first generate initial prey-predator proportions $(x, y) \sim \mathcal{U}[0, 1]^2$ and compute the resulting initial states $\mathbf{x}_0 = (\ln(x), \ln(y))$. We set $\alpha = \beta = \gamma = \delta = 1$ and generate trajectories by integrating for 400 steps ($\delta t = 0.05$) using a 4th order Runge-Kutta scheme, with an
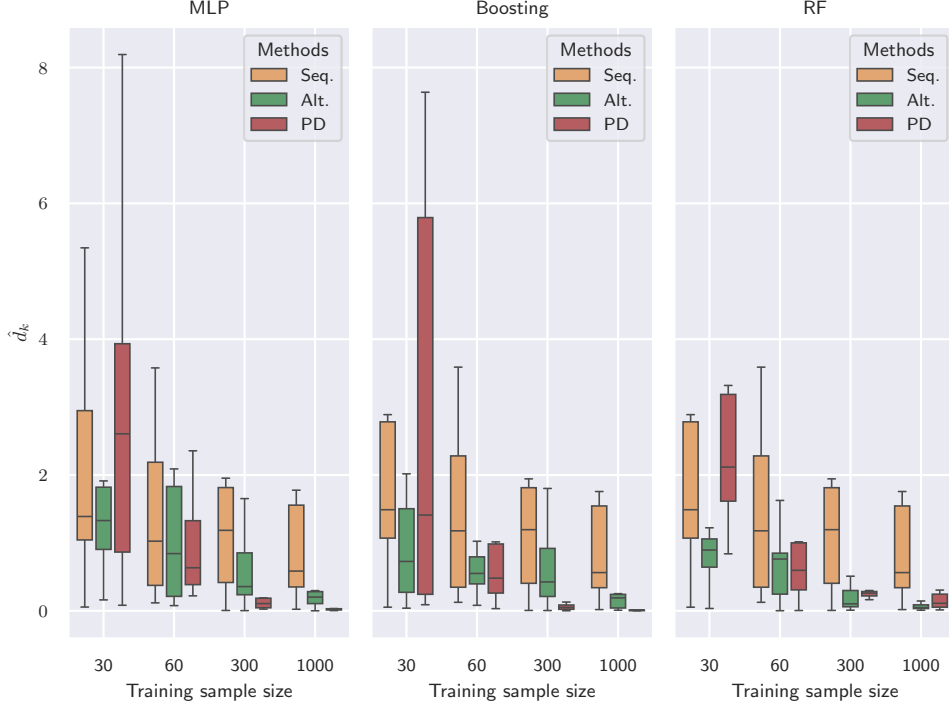
Fig. C2 Evolution of $\hat{d}_k$ w.r.t. the training sample size, on the *Correlated Friedman* problem. The boxplot summarizes the results over 10 test sets.

integration timestep $\delta t_{sim} = 0.001$. We split the dataset into train-validation-test subsets (50/25/25%). The train set is eventually obtained by extracting windows of 41 timesteps (with a stride of 2) from the original train split, yielding 18 100 smaller sub-trajectories.

For the data-driven, joint and alternate schemes, models are trained for 500 epochs using the Euler integrator with $\delta t = 0.05$. For the PD-based optimization scheme, each training consists of 50 epochs repeated 10 times, with the same integration settings. The final training of h_a consists of 150 epochs. All methods are optimized with Adam ([Kingma and Ba, 2014](#)).

D.2   *Damped Pendulum* system

The damped pendulum dynamics are described by:

$$
\begin{cases}
\dot{\theta} & = \frac{d\theta}{dt} \\
\ddot{\theta} & = -\omega_0^2 \sin\theta - \xi\frac{d\theta}{dt}
\end{cases}
$$

where $\mathbf{x} = (\theta, \frac{d\theta}{dt})$ is the system state.

We simulate a dataset of 200 trajectories spanning 201 timesteps (including the initial state $\mathbf{x}_0$), over a time interval of $[0, 10]$s. We first generate the initial states

$\mathbf{x}_0 = (\theta_0, \frac{d\theta_0}{dt}) \sim \mathcal{U}[\frac{-\pi}{2}, \frac{pi}{2}] \times \mathcal{U}[0, 0.1]$. We set $(\omega, \xi) \sim \mathcal{U}[0.785, 3.14] \times \mathcal{U}[0, 0.8]$ and generate trajectories by integrating for 200 steps ($\delta t = 0.05$) using a 4th order Runge-Kutta scheme, with an integration timestep $\delta t_{sim} = 0.001$. We split the dataset into train-validation-test subsets (50/25/25%). The train set is eventually obtained by extracting windows of 41 timesteps (with a stride of 2) from the original train split, yielding 8 100 smaller sub-trajectories.

For the data-driven, joint and alternate schemes, models are trained for 500 epochs using the Euler integrator with $\delta t = 0.05$. For the PD-based optimization scheme, each training consists of 50 epochs repeated 10 times, with the same integration settings. The final training of h_a consists of 150 epochs. All methods are optimized with Adam (Kingma and Ba, 2014).

D.3 *Reaction-Diffusion* system

The reaction-diffusion system is a 2-dimensional system driven by the following PDEs (Klaasen and Troy, 1984):

$$\begin{cases} \frac{\delta u}{\delta t} &= \alpha \Delta u + R_u(u, v; \gamma) \\ \frac{\delta v}{\delta t} &= \beta \Delta v + R_v(u, v) \end{cases}$$

where $\mathbf{x} = (u, v)$ is the system state. $R_u(u, v; \gamma) = u - u^3 - \gamma - v$ and $R_v(u, v) = u - v$ are the local reaction terms. We choose a rectangular domain spanning $[-1, 1]^2$ with circular boundary conditions, that we discretize by a 32×32 uniform grid $G(t)$, where Δ is the 3×3 Laplace operator. For this system, (u, v) values are associated to each of the 32×32 grid cells G_{ij} that are governed by the same PDEs. Therefore, we decided to model $f_a = (R_u(u, v; \gamma), R_v(u, v))$ through a CNN with input $G(t)$.

For the generation of the dataset, we take inspiration from the work of Yin et al. (2021). We simulate a dataset of 1 920 trajectories spanning 246 timesteps (including the initial state $\mathbf{x}_0$), over a time interval of $[0, 2.45]$s. We first generate the initial states $\mathbf{x}_0 = (u_0, v_0) \sim \mathcal{U}[0, 1]^2$. We set $\alpha = 1\mathrm{e}{-3}, \beta = 5\mathrm{e}{-3}, \gamma = 5\mathrm{e}{-3}$ and generate trajectories by integrating for 245 steps ($\delta t = 0.01$) using a 4th order Runge-Kutta scheme, with an integration timestep $\delta t_{sim} = 0.001$. We split the dataset into train-validation-test subsets (50/25/25%). The train set is eventually obtained by extracting windows of 51 timesteps (with a stride of 20) from the original train split, yielding 9 600 smaller sub-trajectories.

For the data-driven, joint and alternate schemes, models are trained for 1 000 epochs using the Euler integrator with $\delta t = 0.01$. For the PD-based optimization scheme, each training consists of 100 epochs repeated 10 times, with the same integration settings. The final training of h_a consists of 1 000 epochs. All methods are optimized with Adam (Kingma and Ba, 2014).

D.4 Dynamical systems forecasting - Additional results

In this section, we report results for variants of the optimization methods presented in Section 4.2. In particular, we test the initialization introduced by Donà et al. (2022) with both joint and PD-based training. For PD-based training, this means replacing

the first two steps of Algorithm 2 with a fit of $(\hat{\theta}_k, \hat{\gamma})$ to minimize $d(h_k^{\theta_k} + \gamma, f)$. We also compare the impact of regularizing h_a for both joint and alternate optimizations.

D.4.1 *Lotka-Volterra* system

Table D6 Results for other *Lotka-Volterra* system, averaged over 5 different initializations of h. Init. refers to $(\hat{\theta}_k, \hat{\gamma}) = \arg\min_{\theta_k, \gamma} d(h_k^{\theta_k} + \gamma, f)$ while Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Methods not included in the table in the main paper are in italics.

Method	log-$\hat{d}(h, f; TS)$	$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$
$f_k \to h_a$	-5.45 ± 0.02	-
Joint - No Init. & No Reg.	-5.22 ± 0.04	$2.51 \times 10^{-1} \pm 3.90 \times 10^{-2}$
Joint - No Init. & Reg.	-5.45 ± 0.06	$1.17 \times 10^{-6} \pm 1.47 \times 10^{-13}$
Joint - Init. & No Reg.	**-5.53 ± 0.01**	$8.62 \times 10^{-2} \pm 2.63 \times 10^{-3}$
Joint - Init. & Reg.	-5.42 ± 0.04	$1.66 \times 10^{-6} \pm 7.63 \times 10^{-13}$
Alternate - No Init. & No Reg.	-5.23 ± 0.09	$5.58 \times 10^{-1} \pm 1.64 \times 10^{-1}$
Alternate - No Init. & Reg.	-5.32 ± 0.01	$1.66 \times 10^{-1} \pm 5.00 \times 10^{-4}$
Alternate - Init. & No Reg.	-5.19 ± 0.04	$3.79 \times 10^{-1} \pm 4.49 \times 10^{-2}$
Alternate - Init. & Reg.	**-5.52 ± 0.05**	$1.35 \times 10^{-1} \pm 7.19 \times 10^{-3}$
PD-based - No Init.	-5.41 ± 0.04	$1.05 \times 10^{-7} \pm 9.51 \times 10^{-15}$
PD-based - Init.	**-5.51 ± 0.02**	$\mathbf{6.27 \times 10^{-8} \pm 8.72 \times 10^{-15}}$
h_a only	-5.02 ± 0.09	-

Compared to Section 4.3.1, Table D6 informs us that, for this problem, PD-based training can yield models equally good to alternate optimization, provided we initialize h_k as specifically defined by Donà et al. (2022). We can also note that the resulting estimation of h_k is even better. Moreover, joint training with no regularization reaches similar predictive performance, at the cost of a worse estimation of h_k (although better than that of alternate training). Given that the parametric component we fit is $h_k^{\theta_k} + \gamma$ (with $\gamma \in \mathbb{R}$), initializing h_k as $(\hat{\theta}_k, \hat{\gamma}) = \arg\min_{\theta_k, \gamma} d(h_k^{\theta_k} + \gamma, f)$ actually makes γ converge to α, which explains why training methods seem to benefit from this initialization in this particular system.

We also investigated the impact of removing this specific h_k initialization from the alternate scheme. For this problem, it appeared clearly that such initialization enhanced the final model h.

D.4.2 *Damped Pendulum* system

Similarly to Section 4.3.2, we can see from Table D7 that PD-based training reaches the highest performance both for trajectory estimation as well as for the estimation of h_k. For this system, the particular h_k initialization does not seem to impact our training method, on any level.

There is now more nuance to the impact of such initialization on joint optimization. Indeed, while it seems to help the non-regularized configuration, the effect is the opposite when we add the presence of their regularizer, indicating that hyperparameters could have been optimized even more or simply that their regularization criterion

Table D7 Results for the *Damped Pendulum* system, averaged over 5 different initializations of h. Init. refers to $(\hat{\theta}_k, \hat{\gamma}) = \arg\min_{\theta_k, \gamma} d(h_k^{\theta_k} + \gamma, f)$ while Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Methods not included in the table in the main paper are in italics.

Method	$\log\text{-}\hat{d}(h, f; TS)$	$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$
$f_k \to h_a$	**-8.75 $\pm$ 0.01**	-
Joint - No Init. & No Reg.	-7.68 ± 0.03	$2.29 \times 10^{-1} \pm 4.47 \times 10^{-3}$
Joint - No Init. & Reg.	-5.98 ± 0.04	$5.51 \times 10^{-3} \pm 2.36 \times 10^{-6}$
Joint - Init. & No Reg.	-7.74 ± 0.02	$1.57 \times 10^{-1} \pm 3.75 \times 10^{-3}$
Joint - Init. & Reg.	-5.83 ± 0.03	$2.83 \times 10^{-2} \pm 2.26 \times 10^{-3}$
Alternate - No Init. & No Reg.	-7.23 ± 0.04	$2.34 \times 10^{-1} \pm 4.47 \times 10^{-3}$
Alternate - No Init. & Reg.	-6.91 ± 0.00	$6.84 \times 10^{-3} \pm 4.56 \times 10^{-6}$
Alternate - Init. & No Reg.	-7.36 ± 0.02	$1.61 \times 10^{-1} \pm 4.09 \times 10^{-3}$
Alternate - Init. & Reg.	-6.90 ± 0.00	$5.90 \times 10^{-3} \pm 2.64 \times 10^{-6}$
PD-based - No Init.	-8.28 ± 0.04	$\mathbf{2.42 \times 10^{-5} \pm 4.57 \times 10^{-12}}$
PD-based - Init.	-8.29 ± 0.02	$\mathbf{2.36 \times 10^{-5} \pm 5.56 \times 10^{-12}}$
h_a only	-7.28 ± 0.01	-

is not suited with this specific initialization. However, we can observe that the effect is positive for models optimized in an alternate fashion.

D.4.3 *Reaction-Diffusion* system

Table D8 Results for the *Reaction-Diffusion* system, averaged over 5 different initializations of h. Init. refers to $(\hat{\theta}_k, \hat{\gamma}) = \arg\min_{\theta_k, \gamma} d(h_k^{\theta_k} + \gamma, f)$ while Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Methods not included in the table in the main paper are in italics.

Method	$\log\text{-}\hat{d}(h, f; TS)$	$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$
$f_k \to h_a$	-3.43 ± 0.11	-
Joint - No Init. & No Reg.	**-3.62 $\pm$ 0.02**	$\mathbf{1.17 \times 10^{-5} \pm 1.42 \times 10^{-10}}$
Joint - No Init. & Reg.	-2.96 ± 0.01	$5.45 \times 10^{-4} \pm 2.64 \times 10^{-7}$
Joint - Init. & No Reg.	-3.14 ± 0.11	$4.43 \times 10^{-5} \pm 2.61 \times 10^{-9}$
Joint - Init. & Reg.	-2.65 ± 0.04	$1.14 \times 10^{-3} \pm 2.50 \times 10^{-7}$
Alternate - No Init. & No Reg.	-3.26 ± 0.02	$1.94 \times 10^{-4} \pm 6.95 \times 10^{-9}$
Alternate - No Init. & Reg.	-2.98 ± 0.02	$1.71 \times 10^{-4} \pm 6.18 \times 10^{-9}$
Alternate - Init. & No Reg.	-2.99 ± 0.02	$4.80 \times 10^{-5} \pm 2.81 \times 10^{-9}$
Alternate - Init. & Reg.	-2.97 ± 0.01	$1.89 \times 10^{-3} \pm 1.67 \times 10^{-6}$
PD-based - No Init.	-3.09 ± 0.02	$1.64 \times 10^{-3} \pm 1.01 \times 10^{-7}$
PD-based - Init.	-3.12 ± 0.02	$2.66 \times 10^{-3} \pm 2.51 \times 10^{-6}$
h_a only	-2.86 ± 0.01	-

For this 2-D problem, Table D8 suggests that non-regularized joint optimization with uniform h_k initialization performs best, as could be seen from Table 9. As was said, PD-based training seems to struggle now that there is a clear overlap between f_k and f_a. Furthermore, the particular h_k initialization does not seem to impact our training method.

As concerns joint optimization, the initialization proposed by Donà et al. (2022) clearly degrades both performance criteria. Nevertheless, we can observe that once

again the effect is positive for regularized alternate training, indicating that, as could be observed from previous systems, there is a very sensitive balance to find between the initialization of the physical component and the regularizers, along with their optimal coefficient values.

Appendix E Training methods pseudo-codes

Algorithm 3 Sequential Optimization: Static problems

Input: LS $= \{(\mathbf{x}_i, y_i)\}_{i=1}^N$

$(\hat{\theta}_k, \hat{\gamma}) \leftarrow \mathrm{fit}^{h_k+\gamma}(LS)$

$\hat{\theta}_a \leftarrow \mathrm{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k}(\mathbf{x}_i) - \hat{\gamma})\}_{i=1}^N)$

return $h_k^{\hat{\theta}_k} + \hat{\gamma} + h_a^{\hat{\theta}_a}$

Algorithm 4 Joint Optimization: Non-Regularized Setting - Dynamical problems

---

Input: LS $= \{(\mathbf{x}_0^{(i)}, \ldots, \mathbf{x}_T^{(i)})\}_{i=1}^N$, $\eta > 0$

$\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}, \hat{\theta}_a^{(0)} \leftarrow$ Random Initialization

for $e = 1$ **to** N_e **do**

$\quad (\hat{\theta}_k^{(e)}, \hat{\gamma}^{(e)}, \hat{\theta}_a^{(e)}) \leftarrow (\hat{\theta}_k^{(e-1)}, \hat{\gamma}^{(e-1)}, \hat{\theta}_a^{(e-1)}) - \eta \nabla_{(\hat{\theta}_k, \hat{\gamma}, \hat{\theta}_a)}[\hat{d}(h_k^{\hat{\theta}_k^{(e-1)}} + \hat{\gamma}^{(e-1)} + h_a^{\hat{\theta}_a^{(e-1)}}, f; LS)]$

end for

return $h_k^{\hat{\theta}_k^{(N_e)}} + \hat{\gamma}^{(N_e)} + h_a^{\hat{\theta}_a^{(N_e)}}$

Algorithm 5 Alternate Optimization: Static problems

Input: LS $= \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $\eta > 0$

$\hat{\theta}_a^{(0)} \leftarrow$ Random Initialization

$(\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}) \leftarrow \mathrm{fit}^{h_k+\gamma}(LS)$

for $e = 1$ **to** N_e **do**

$\quad \hat{\theta}_a^{(e)} \leftarrow \mathrm{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k^{(e-1)}}(\mathbf{x}_i) - \hat{\gamma}^{(e-1)})\}_{i=1}^N)$

$\quad (\hat{\theta}_k^{(e)}, \hat{\gamma}^{(e)}) \leftarrow (\hat{\theta}_k^{(e-1)}, \hat{\gamma}^{(e-1)}) - \eta \nabla_{(\hat{\theta}_k, \hat{\gamma})}[\hat{d}(h_k^{\hat{\theta}_k^{(e-1)}} + \hat{\gamma}^{(e-1)} + h_a^{\hat{\theta}_a^{(e)}}, y; LS)]$

end for

$\hat{\theta}_a^{(N_e+1)} \leftarrow \mathrm{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k^{(N_e)}}(\mathbf{x}_i) - \hat{\gamma}^{(N_e)})\}_{i=1}^N)$

return $h_k^{\hat{\theta}_k^{(N_e)}} + \hat{\gamma}^{(N_e)} + h_a^{\hat{\theta}_a^{(N_e+1)}}$

Algorithm 6 Alternate Optimization: Non-Regularized Setting - Dynamical problems
---
Input: LS $= \{(\mathbf{x}_0^{(i)}, \ldots, \mathbf{x}_T^{(i)})\}_{i=1}^N$, $\eta > 0$

$\hat{\theta}_a^{(0)} \leftarrow$ Random Initialization
$(\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}) \leftarrow \text{fit}_{\text{NODE}}^{h_k + \gamma}(LS)$
for $e = 1$ **to** N_e **do**
$\qquad \hat{\theta}_a^{(e)} \leftarrow \hat{\theta}_a^{(e-1)} - \eta \nabla_{\hat{\theta}_a}[\hat{d}(h_a^{\hat{\theta}_a^{(e-1)}} + h_k^{\hat{\theta}_k^{(e-1)}} + \hat{\gamma}^{(e-1)}, f; LS)]$
$\qquad (\hat{\theta}_k^{(e)}, \hat{\gamma}^{(e)}) \leftarrow (\hat{\theta}_k^{(e-1)}, \hat{\gamma}^{(e-1)}) - \eta \nabla_{(\hat{\theta}_k, \hat{\gamma})}[\hat{d}(h_k^{\hat{\theta}_k^{(e-1)}} + \hat{\gamma}^{(e-1)} + h_a^{\hat{\theta}_a^{(e)}}, f; LS)]$
end for
$\hat{\theta}_a^{(N_e+1)} \leftarrow \hat{\theta}_a^{(N_e)} - \eta \nabla_{\hat{\theta}_a}[\hat{d}(h_a^{\hat{\theta}_a^{(N_e)}} + h_k^{\hat{\theta}_k^{(N_e)}} + \hat{\gamma}^{(N_e)}, f; LS)]$
return $h_k^{\hat{\theta}_k^{(N_e)}} + \hat{\gamma}^{(N_e)} + h_a^{\hat{\theta}_a^{(N_e+1)}}$