Learning interacting dynamical systems with latent Gaussian process ODEs

Çağatay Yıldız*
University of Tübingen
cagatay.yildiz@uni-tuebingen.de

Melih Kandemir
University of Southern Denmark
kandemir@imada.sdu.dk

Barbara Rakitsch
Bosch Center for Artificial Intelligence
barbara.rakitsch@de.bosch.com

Abstract

We study uncertainty-aware modeling of continuous-time dynamics of interacting objects. We introduce a new model that decomposes independent dynamics of single objects accurately from their interactions. By employing latent Gaussian process ordinary differential equations, our model infers both independent dynamics and their interactions with reliable uncertainty estimates. In our formulation, each object is represented as a graph node and interactions are modeled by accumulating the messages coming from neighboring objects. We show that efficient inference of such a complex network of variables is possible with modern variational sparse Gaussian process inference techniques. We empirically demonstrate that our model improves the reliability of long-term predictions over neural network based alternatives and it successfully handles missing dynamic or static information. Furthermore, we observe that only our model can successfully encapsulate independent dynamics and interaction information in distinct functions and show the benefit from this disentanglement in extrapolation scenarios.

1 Introduction

A broad spectrum of dynamical systems consists of multiple interacting objects. Since their interplay is typically *a priori* unknown, learning interaction dynamics of objects from data has become an emerging field in dynamical systems [1, 2, 3]. The ever-growing interest in interaction modeling is due to the diversity of real-world applications such as autonomous driving [4], physical simulators [5], and human-robot interactions [6]. Standard time-series algorithms or deep learning approaches (e.g. recurrent neural networks), that have been designed for single-object systems, do not scale to a large number of interacting objects since they do not exploit the structural information of the data.

In recent years, graph neural networks (GNNs) have emerged as a promising tool for interactive systems, where objects are represented as graph nodes. State-of-the-art methods learn interactions by sending messages between objects in form of multi-layer perceptrons [1] or attention modules [7]. These methods yield highly flexible function approximators that achieve accurate predictions when trained on large-scale datasets. However, their predictions come without calibrated uncertainties, hindering their reliable implementation for uncertainty-aware applications.

In contrast, Gaussian processes (GPs) are well-known for providing calibrated uncertainty estimates. They have been successfully employed on discrete time-series data [8, 9, 10] and, more recently,

*Work performed during internship at Bosch Center for Artificial Intelligence

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

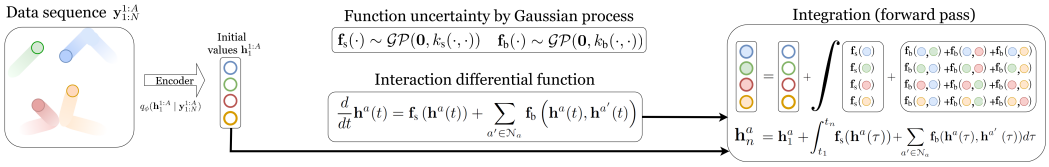

Figure 1: A high-level overview of our I-GPODE model. *(Left)* A bouncing balls sequence with four balls in a box. The balls move independently other than collision (interaction) times. An encoder is used to extract initial values. *(Middle)* GP priors on the independent kinematics and interaction functions, which form together the differential function. *(Right)* Given the initial value and function samples, object trajectories are computed using the forward integration of the differential function.

to continuous-time generalizations of these methods [11, 12, 13]. Importantly, none of the these works adresses dynamical models for interacting systems. While it is possible to study each object in isolation, ignoring the interaction effects might lead to inaccurate predictions.

In this work, we address the shortcomings of both model families by presenting an uncertainty-aware continuous-time dynamical model for interacting objects. Our formulation decomposes the dynamics into independent (autonomous) and interaction dynamics. While the former governs the motion of an object in isolation, the latter describes the effects that result from interactions with neighboring objects. For successful uncertainty characterization, we propose to infer the unknown independent and interaction dynamics by two distinct GPs. We demonstrate that having a function-level GP prior on the individual dynamics components is the key to successfully disentangling these dynamics, which in turn allows for interpretable predictions and leads to improved extrapolation behavior.

We employ latent Gaussian process ordinary differential equations (GP-ODEs) for dynamics learning, allowing to learn complex relationships between interacting objects without the need of having access to fully observed systems. Thanks to recently proposed decoupled sampling scheme [14], the computational complexity of our model scales linearly with the number of time points at which the ODE system is evaluated. As a result, our algorithm scales gracefully to datasets with thousands of sequences. To demonstrate the benefits of our framework, we exhaustively test our method on a wide range of scenarios varying in function complexity, signal-to-noise ratio, and system observability. Our model consistently outperforms non-interacting dynamical systems and alternative function approximators such as deterministic/Bayesian neural networks.

2 Background

In this section, we give background on continuous-time systems and Gaussian processes. Both together form the backbone of our uncertainty-aware framework for interactive dynamical systems.

2.1 Continuous-time Dynamical Systems

Continuous-time dynamical systems are often expressed using differential functions $\dot{\mathbf{x}}(t) \equiv \frac{d}{dt}\mathbf{x}(t) \equiv \mathbf{f}(\mathbf{x}(t))$, where $\mathbf{x}(t) \in \mathbb{R}^D$ represents the state of an ODE system at time t and and $\mathbf{f} : \mathbb{R}^D \mapsto \mathbb{R}^D$ is the *time differential function* governing the dynamics evolution. The state solution $\mathbf{x}(t_1)$ at an arbitrary time t_1 is characterized by the initial value at time point t_0 and the differential function:

$$\mathbf{x}(t_1) = \mathbf{x}(t_0) + \int_{t_0}^{t_1} \mathbf{f}(\mathbf{x}(\tau))\, d\tau.$$

Existing work aims to approximate the unknown differential by Gaussian processes [15, 11] or neural networks [16]. These methods have shown to accurately capture the dynamics and outperform their discrete-time counterparts in a wide range of applications such as time series forecasting [17], classification [18] or reinforcement learning [19]. Furthermore, ODE models allow to easily inject domain knowledge into the system, enabling interpretable and flexible hybrid models [20, 21, 22].

2.2 Gaussian Processes

Gaussian processes (GPs) define priors over functions [23]:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')),$$

where $f : \mathbb{R}^D \to \mathbb{R}$ maps D-dimensional inputs into one-dimensional outputs. GPs are fully specified in terms of their mean and their covariance:

$$\mathbb{E}[f(\mathbf{x})] = \mu(\mathbf{x}), \qquad \mathrm{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}'),$$

where $\mu : \mathbb{R}^D \to \mathbb{R}$ is the mean and $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the kernel function. GPs can be treated as an extension of a multivariate normal distribution to infinitely many dimensions, where any fixed set of inputs $\mathbf{X} \in \mathbb{R}^{N \times D}$ follows the Gaussian distribution

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}_{\mathbf{X}}, \mathbf{K}_{\mathbf{XX}}), \tag{1}$$

with $\boldsymbol{\mu}_{\mathbf{X}}$ being the mean function evaluated at inputs $\mathbf{X}$, and $\mathbf{K}_{\mathbf{XX}}$ the kernel function evaluated at all input pairs in $\mathbf{X}$. While GPs provide a natural mechanism to handle uncertainties, their computational complexity grows cubically with the number of input points. This problem is often tackled by *sparse GPs*, which rely on augmenting the GP with M inducing inputs $\mathbf{Z} = [\mathbf{z}_1^T, \ldots, \mathbf{z}_M^T]^T \in \mathbb{R}^{M \times D}$ and corresponding output variables $\mathbf{u} = [u_1, \ldots, u_M]^T \in \mathbb{R}^{M \times 1}$ with $u_m \equiv f(\mathbf{z}_m)$ [24, 25]. Assuming a zero-mean prior, as it is common in the literature, the conditional distribution over $f(\mathbf{X})$ follows the Gaussian process:

$$p(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}(\mathbf{f} \mid \mathbf{K}_{\mathbf{XZ}}\mathbf{K}_{\mathbf{ZZ}}^{-1}\mathbf{u},\ \mathbf{K}_{\mathbf{XX}} - \mathbf{K}_{\mathbf{XZ}}\mathbf{K}_{\mathbf{ZZ}}^{-1}\mathbf{K}_{\mathbf{ZX}}), \tag{2}$$

where $\mathbf{K}_{\mathbf{ZZ}}$ is the covariance between all inducing points $\mathbf{Z}$, and $\mathbf{K}_{\mathbf{XZ}}$ between the input points $\mathbf{X}$ and the inducing points $\mathbf{Z}$. The inducing points can thereby be interpreted as a compressed version of the training data in which the number of inducing points M acts as a trade-off parameter between the goodness of the approximation and scalability.

In this work, we employ the squared exponential kernel $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D} \frac{(x_d - x_d')^2}{\ell_d^2}\right)$, where x_d denotes the d-th entry of the input $\mathbf{x}$, σ^2 is the output variance and ℓ_d is the dimension-wise lengthscale parameter.

3 Interacting Dynamical Systems with Latent Gaussian Process ODEs

In Sec. 3.1, we describe our continuous-time formulation for systems of interacting objects. It decomposes the dynamics into independent kinematics and an interaction component that takes the interactions to neighboring objects into account. Placing a GP prior over the individual components is essential in order to arrive at *(i)* calibrated uncertainty estimates and *(ii)* disentangled representations as we later on also verify in our experiments. In Sec. 3.2, we embed the GP dynamics into a latent space that can accomodate missing static or dynamic information. Both together allows the application of our continuous-time formulation to a wide range of scenarios and allows for learning interpretable dynamics. We conclude this section by our variational inference framework (Sec 3.3) based on sampling functions from GP posteriors.

3.1 Interacting Dynamical Systems

We assume a dataset of P sequences $\mathcal{Y} = \{\mathbb{Y}_1, \ldots, \mathbb{Y}_P\}$, where each sequence $\mathbb{Y} \equiv \mathbf{Y}_{1:N} \equiv \mathbf{y}_{1:N}^{1:A}$ is composed of measurements of A objects at time points $\mathcal{T} = \{t_1, \ldots, t_N\}$. Without loss of generality, we assume that the measurement $\mathbf{y}^a(t_n) \equiv \mathbf{y}_n^a \in \mathbb{R}^O$ is related to the physical properties of object a, such as position and velocity, which can routinely be measured by standard sensors. The *dynamic state* of object a at any arbitrary time t is denoted by a latent vector $\mathbf{h}^a(t) \in \mathbb{R}^D$, which does not necessarily live in the same space as the observations. We furthermore assume that each object a is associated with a *global* feature vector $\mathbf{c} \in \mathbb{R}^C$, which corresponds to the static attributes that remain constant over time. Finally, we denote the concatenation of all states by $\mathbf{H}(t) = [\mathbf{h}^1(t), \ldots \mathbf{h}^A(t)] \in \mathbb{R}^{A \times D}$ and all globals by $\mathbf{C} = [\mathbf{c}^1, \ldots \mathbf{c}^A] \in \mathbb{R}^{A \times C}$.

In the following, we propose to disentangle the complex continuous-time dynamics into *independent kinematics* and *interaction* differentials. More concretely, we introduce the following dynamics:

$$\frac{d}{dt}\mathbf{H}(t) = \left[\frac{d}{dt}\mathbf{h}^1(t), \ldots, \frac{d}{dt}\mathbf{h}^A(t)\right], \tag{3}$$

$$\frac{d}{dt}\mathbf{h}^a(t) = \mathbf{f}_{\mathrm{s}}\left(\mathbf{h}^a(t), \mathbf{c}^a\right) + \sum_{a' \in \mathcal{N}_a} \mathbf{f}_{\mathrm{b}}\left(\mathbf{h}^a(t), \mathbf{h}^{a'}(t), \mathbf{c}^a, \mathbf{c}^{a'}\right), \tag{4}$$

where $\mathcal{N}_a$ denotes the set of neighbors of object a in a given graph. The first function $\mathbf{f}_\text{s} : \mathbb{R}^{D+C} \mapsto \mathbb{R}^D$ models the independent (autonomous) effects, which specifies how the object would behave without any interactions. The second function $\mathbf{f}_\text{b} : \mathbb{R}^{2D+2C} \mapsto \mathbb{R}^D$ models the *interactions* by accumulating messages coming from all neighboring objects. Since message accumulation is the *de-facto* choice in interaction modeling [2, 26], the additive form of the differential function is a very generic inductive bias.

Furthermore, our formulation models interactions between pairs of objects explicitly via the differential equation (Eq. (5)). Higher-order interactions are taken into account via the continuous-time formulation that allows information to propagate through the complete graph via local messages over time such that the state of the object $\mathbf{h}_n^a$ can also depend on objects that are not directly connected in the graph. In contrast to discrete formulations, for which the message passing speed is limited by the sampling rate, our continuous-time formulation enjoys instant propagation of information across objects. Finally, please see Section for an investigation of our interaction component under a kernel perspective.

Remark-1 In Sec A.1 we demonstrate two straightforward extensions of our formulation with non-linear message accumulation, which we empirically show to have no improvement over our proposed, simpler formulation.

Remark-2 Previous GP-based ODE methods [15, 11, 12] assume a black-box approximation to the unknown system $\frac{d}{dt}\mathbf{H}(t) = \mathbf{f}(\mathbf{H}(t))$ whereas our state representation gracefully scales to a varying number of objects and also include global features.

3.2 Probabilistic Generative Model

Real-world data of interactive systems necessitates embedding the dynamics into a latent space in order to allow for missing information; the observations may contain only partial estimates of the states and the globals $\mathbf{C}$ might not be observed at all. We account for those circumstances by treating the states $\mathbf{H}$ and the globals $\mathbf{C}$ as latent variables, leading to following generative model (see Figure 1):

$$
\begin{aligned}
&\mathbf{h}_1^a \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{c}^a \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\
&\mathbf{f}_\text{s}(\cdot) \sim \mathcal{GP}(\mathbf{0}, k_\text{s}(\cdot,\cdot)), \quad \mathbf{f}_\text{b}(\cdot) \sim \mathcal{GP}(\mathbf{0}, k_\text{b}(\cdot,\cdot)), \\
&\mathbf{h}_n^a = \mathbf{h}_1^a + \int_{t_1}^{t_n} \mathbf{f}_\text{s}\left(\mathbf{h}^a(\tau), \mathbf{c}^a\right) + \sum_{a' \in \mathcal{N}_a} \mathbf{f}_\text{b}\left(\mathbf{h}^a(\tau), \mathbf{h}^{a'}(\tau), \mathbf{c}^a, \mathbf{c}^{a'}\right) d\tau, \\
&\mathbf{y}_n^a \sim p(\mathbf{y}_n^a | \mathbf{h}_n^a),
\end{aligned}
\tag{5}
$$

where we introduced a standard Gaussian prior over the initial latent state, and assumed that the data likelihood decomposes across time and objects. We furthermore model unknown functions $\mathbf{f}_\text{s}$ and $\mathbf{f}_\text{b}$ under independent vector-valued GP priors.

In our experiments, we further set $p(\mathbf{y}_n^a|\mathbf{h}_n^a) = \mathcal{N}(\mathbf{y}_n^a|\mathbf{B}\mathbf{h}_n^a, \text{diag}(\boldsymbol{\sigma}_e^2))$, where $\mathbf{B} \in \mathbb{R}^{O \times D}$ maps from the latent to the observational space and $\boldsymbol{\sigma}_e^2 \in \mathbb{R}_+^O$ is the noise variance. We further fix $\mathbf{B} = [\mathbf{I}, \mathbf{0}]$ where $\mathbf{I} \in \mathbb{R}^{O \times O}, \mathbf{0} \in \mathbb{R}^{O, D-O}$, in order to arrive at an interpretable latent space in which the first dimensions correspond to the observables. This assumption is fairly standard in the GP state-space model literature since more complex emission models can be subsumed in the transition model without reducing the model complexity [27].

Modeling partially observed systems often leads to non-identifiability issues that hamper optimization and ultimately lead to deteriorated generalization performance. One way to counteract this behavior is to inject prior physical knowledge into the system by decomposing the state space of each object $\mathbf{h}^a(t) \equiv [\mathbf{s}^a(t),\ \mathbf{v}^a(t)]$ into *position* $\mathbf{s}^a(t)$ and *velocity* $\mathbf{v}^a(t)$ components [28]. Using elementary physics, the differential function has then the form of $\frac{d}{dt}\mathbf{h}^a(t) = \left[\mathbf{v}^a(t), \frac{d}{dt}\mathbf{v}^a(t)\right]$ with $\frac{d}{dt}\mathbf{v}^a(t) = \mathbf{f}_\text{s}\left(\mathbf{h}^a(t), \mathbf{c}^a\right) + \sum_{a' \in \mathcal{N}_a} \mathbf{f}_\text{b}\left(\mathbf{h}^a(t), \mathbf{h}^{a'}(t), \mathbf{c}^a, \mathbf{c}^{a'}\right)$.

Remark Unlike previous work, our formulation incorporates global features $\mathbf{c}$ that modulate the dynamics. In many applications such as control engineering and reinforcement learning, the dynamics are modulated by external control signals [29] which can also be incorporated into our framework.

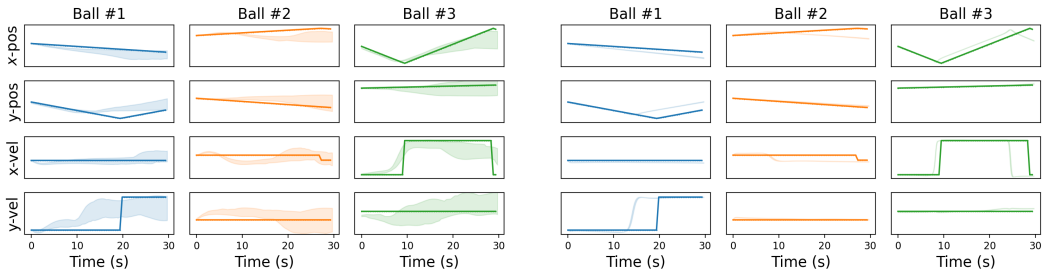

Figure 2: Test predictions for I-GPODE (left) and I-NODE (right) on bouncing balls dataset. The solid curves are the groundtruth trajectories and the shaded regions denote the predicted 95% confidence intervals. I-GPODE (ours) yields better calibrated long-term predictions than I-NODE. Additional results can be found in Figure 5.

3.3 Variational Inference

Next, we derive an efficient approximate inference scheme that provides a high level of accuracy and model flexibility. In the above formulation, the model unknowns are the initial values $\mathbf{H}_1 \equiv \mathbf{h}_1^{1:A}$, global variables $\mathbf{C} = \mathbf{c}^{1:A}$ and the differentials $\mathbf{f}_\text{s}$ and $\mathbf{f}_\text{b}$. Since the exact posterior $p(\mathbf{f}_\text{s}, \mathbf{f}_\text{b}, \mathbf{H}_1\ \mathbf{C}|\mathcal{Y})$ in non-linear ODE models is intractable, we opt for stochastic variational inference [30]. We first describe the form of the approximate posterior and then discuss how to optimize its parameters.

Variational family Similarly to previous work [16], we resort to amortized inference for initial values and global variables, $\mathbf{H}_1 \sim q_\phi(\mathbf{H}_1|\mathbf{Y}_{1:N})$ and , $\mathbf{C} \sim q_\rho(\mathbf{C}|\mathbf{Y}_{1:N})$, where ϕ is the parameters of a neural network encoder that outputs a Gaussian distribution with diagonal covariance. For the unknown time differentials $\mathbf{f} = \{\mathbf{f}_\text{s}, \mathbf{f}_\text{b}\}$, we follow the standard sparse GP approximation in which each output dimension $d \in [1, D]$ has its own independent set of inducing values $\mathbf{U}_{s,d}, \mathbf{U}_{b,d} \in \mathbb{R}^D$ and kernel output variances $\sigma^2_{s,d}, \sigma^2_{b,d} \in \mathbb{R}_+$. Inducing locations $\{\mathbf{Z}_s, \mathbf{Z}_d\}$ and lengthscales $\{\boldsymbol{\ell}_\text{s}, \boldsymbol{\ell}_\text{b}\}$ are shared across output dimensions. Denoting the collection of inducing points by $\mathbf{U} = \{\mathbf{U}_\text{s}, \mathbf{U}_\text{b}\}$, we arrive at the mean-field variational posterior:

$$q(\mathbf{U}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{U}_{s,d} \mid \mathbf{m}_{s,d}, \mathbf{S}_{s,d})\mathcal{N}(\mathbf{U}_{b,d} \mid \mathbf{m}_{b,d}, \mathbf{S}_{b,d}),$$

where the means $\{\mathbf{m}_{s,d}, \mathbf{m}_{b,d}\}_{d=1}^D$ and the covariances $\{\mathbf{S}_{s,d}, \mathbf{S}_{b,d}\}_{d=1}^D$ are free variational parameters. Putting everything together, our variational approximation becomes as follows [31]:

$$q(\mathbf{H}_1, \mathbf{C}, \mathbf{f}, \mathbf{U}) \equiv q(\mathbf{H}_1)q(\mathbf{C})p(\mathbf{f}_\text{s}|\mathbf{U}_\text{s})p(\mathbf{f}_\text{b}|\mathbf{U}_\text{b})q(\mathbf{U}),$$

where $p(\mathbf{f}_\text{s}|\mathbf{U}_\text{s})$ and $p(\mathbf{f}_\text{b}|\mathbf{U}_\text{b})$ follow Eq. (2). Our variational family makes two assumptions that are fairly standard in the (deep) GP literature (e.g. [32]): *(i)* we apply the same independence assumptions in the approximate posterior as in the prior resulting in a mean-field solution, and *(ii)* we assume that the inducing outputs $\mathbf{U}$ capture the sufficient statistics of the training data allowing the use of the prior $p(\mathbf{f} \mid \mathbf{U})$ in the approximate posterior.

Variational bound We then seek to optimize the parameters of the approximate posterior q by maximizing a lower bound to the evidence [30]:

$$\log p(\mathcal{Y}) \geq \int q(\mathbf{H}_1, \mathbf{C}, \mathbf{f}, \mathbf{U}) \log \frac{p(\mathcal{Y}, \mathbf{H}_1, \mathbf{C}, \mathbf{f}, \mathbf{U})}{q(\mathbf{H}_1, \mathbf{C}, \mathbf{f}, \mathbf{U})} d\mathbf{H}_1 d\mathbf{C} d\mathbf{f} d\mathbf{U}.$$

In the following, we detail its computation for a single data instance $\mathbf{Y}_{1:N}$, omitting its generalization to multiple sequences for the sake of better readability,

$$\log p(\mathbf{Y}_{1:N}) \geq \mathbb{E}_q\left[\log p(\mathbf{Y}_{1:N}|\mathbf{H}_1, \mathbf{C}, \mathbf{U})\right] - \text{KL}[q(\mathbf{H}_1)||p(\mathbf{H}_1)] - \text{KL}[q(\mathbf{C})||p(\mathbf{C})] - \text{KL}[q(\mathbf{U})||p(\mathbf{U})], \tag{6}$$

where KL denotes the Kullback–Leibler divergence.

Likelihood computation via decoupled sampling from GP posteriors Computing the conditional log-likelihood $\log p(\mathbf{Y}_{1:N}|\mathbf{H}_1, \mathbf{C}, \mathbf{f}, \mathbf{U})$ entails a forward pass in time (Eq. (5)) which can

be done with any standard ODE solver. The difficulty lies in marginalizing over the approximate posterior of the initial latent states $q(\mathbf{H}_1)$, global variables $q(\mathbf{C})$, and the GP functions $q(\mathbf{f}, \mathbf{U})$. Each marginalization step alone is already analytically intractable, let alone their combination. We therefore opt for Monte Carlo integration which gives us an unbiased estimate of the expected log-likelihood. We start by drawing L samples from the approximate posteriors

$$\mathbf{H}_1^{(l)} \sim q_\phi(\mathbf{H}_1|\mathbf{Y}_{1:N}), \quad \mathbf{C}^{(l)} \sim q_\rho(\mathbf{C}|\mathbf{Y}_{1:N}), \quad \mathbf{U}^{(l)} \sim q(\mathbf{U}), \quad \mathbf{f}^{(l)}(\cdot) \sim p(\mathbf{f}|\mathbf{U}), \tag{7}$$

where l denotes the sample index and $\mathbf{f}^{(l)}(\cdot)$ is a function drawn from the sparse GP posterior. Sampling from the GP posterior naively, scales cubically with the number of data points. Moreover, since we do not know a-priori on which points the ODE solver evaluates the function, we would have to sequentially draw points from the posterior. While this can still be done cubically in time by performing low-rank updates, it often leads to numerical instabilities for small step sizes. To overcome this challenge, we resort to the decoupled sampling scheme proposed in [14], where we first draw the prior samples from a set of random Fourier features and then update them using Matheron's rule to obtain posterior samples.

After having sampled the quadruple via Eq. (7), we can compute the trajectory $\mathbf{H}_{1:N}^{(l)}$ deterministically by forward integration of Eq. (5) and the Monte Carlo estimate of the log-likelihood becomes

$$\mathbb{E}_q\left[\log p(\mathbf{Y}_{1:N}|\mathbf{H}_1, \mathbf{C}, \mathbf{f}, \mathbf{U})\right] \approx \frac{1}{L} \sum_{l,n,a} \log p(\mathbf{y}_n^a|\mathbf{h}_n^{a^{(l)}}),$$

where the log-likelihood term decomposes between objects and between time points, enabling doubly stochastic variational inference [33].

KL regularizer The prior distributions over the inducing variables follow the Gaussian process $p(\mathbf{U}) = p(\mathbf{U}_\mathrm{s})p(\mathbf{U}_\mathrm{b})$ with $p(\mathbf{U}_\mathrm{s})$ and $p(\mathbf{U}_\mathrm{b})$ following the equivalent of Eq. (1). Since we furthermore assumed a standard Gaussian prior $p(\mathbf{H}_1) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $p(\mathbf{C}1) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ with suitable dimensions for the initial values, all KL terms can be computed in closed form.

Computational complexity Evaluating the differential function costs $O(M^2(A + I))$ for a graph with A objects, I interactions and M inducing points. The term $O(M^2A)$ stems from the independent kinematics, the term $O(M^2I)$ from the interaction effects. As a consequence, forward integration using RK4 solver takes $O(TM^2(A + I))$ time, where T is the number of time points.

3.4 Comparison to Standard GPODEs

Our approach enhances the capabilities of the GPODE model family in the following three aspects:

Explicit modeling of interactions Standard GPODEs model interactions by allowing the time differential to take the whole state vector $\mathbf{H}(t)$ as input and to learn one independent GP for each latent dimension. This shared latent space assumption entails three major drawbacks: *(i)* obligation to fix the object count as a model hyperparameter, *(ii)* dependency of the learned model on a predefined ordering of the objects in the scene, *(iii)* inevitable growth of the latent dimensionality proportional to the object count. The latter sets a severe bottleneck especially for GP modeling as the performance of many kernel functions in widespread use are highly sensitive to input dimensionality. (For example, on the bouncing ball dataset with N_a balls and D latent states per object, GPODE needs to learn a latent function with N_aD-dimensional inputs and outputs.) In contrast, I-GPODE needs only to learn two functions, the independent kinematics $\mathbf{f}_s$ and the interaction function $\mathbf{f}_b$, whose input sizes scale independently of N_a. Our Table 2 indicates that learning interaction dynamics without the strong inductive bias of our model is difficult and the GPODE model chooses to stay at the prior instead leading to deteriorated MSEs and ELLs.

Disentangled representation We infer object-specific latent variables that modulate the dynamics, which allows our model to disentangle the dynamics evolution (time differential function) from static object properties (e.g. charge information). The interpretability and likely physical correspondence of the disentangled factors have the potential to facilitate the use of our approach in transfer learning, as well as explainable AI applications.

Inference of latent state dynamics We perform the learning in a latent space, where the initial value of a trajectory is given by an encoder, leading to the following two advantages: First, the dynamical system and the data points may live in different spaces, which facilitates learning from

Table 1: A taxonomy of the state-of-the-art ODE systems with NN/GP dynamics, with/without interactions, and with/without latent dynamics. These methods also form the baselines we compare with [neural ODE (NODE), Bayesian NODE (BNODE), latent dynamics (L), interacting (I)].

Function uncertainty	Interacting	Latents	Reference idea	Abbreviation
✗	✗	✗	Neural ODE [16]	NODE
✗	✗	✓	Latent NODE [17]	NODE-L
✗	✓	✗	GDE [34]	I-NODE
✗	✓	✓	LG-ODE [35]	I-NODE-L
✓	✗	✗	Bayesian ODE [11]	GPODE
✓	✗	✓	GP-SDE [13]	GPODE-L
✓	✗	✓	ODE^{2}VAE [28]	BNODE-L
✓	✓	✗	Our work	I-GPODE
✓	✓	✓	Our work	I-GPODE-L

high-dimensional sequences. Second, Bayesian modeling of state dynamics on a latent space enables reliable quantification and principled treatment of sources of uncertainty, such as imprecision of modeling assumptions, approximation error, and measurement noise.

4 Related Work

GPs for ODEs GPs for modeling ODE systems have been studied in a number of publications (e.g. [11, 12, 13, 15, 36]). With the notable exception of [13], they only consider systems in which the dynamics are defined in the data space. The work that is closest to ours from a technical perspective is [11, 12] that also employ decoupled sampling in order to compute consistent trajectories during inference. We are not aware that interacting dynamical systems under a GP prior have been studied previously, either in the continuous or in the discrete time setting.

NODEs for dynamical data Since the debut of Neural ODEs [16], much progress has been made on how to apply neural networks for modeling sequential data in the continuous time domain. These works [17, 37] typically assume a black-box approximation to unknown ODE systems $\frac{d}{dt}\mathbf{H}(t) = \mathbf{f}(\mathbf{H}(t))$, where $\mathbf{f}$ is a deterministic neural network, and the latent space is separated from the observational space using a neural network decoder. A subtle difference to our approach is that we decided to use a linear mapping instead. However, in case that the outputs are high-dimensional, e.g. video data, this can be easily changed. Few works have integrated function uncertainty into Neural ODEs by putting a prior over the neural network weights [28, 38]. To the best of our knowledge, none of these works addressed interacting systems. However, we still compare against an interactive adaptation of these approaches in our experiments.

Modeling interacting dynamics Interacting dynamical systems have first been considered for discrete time-step models and for the deterministic setting [1, 26, 39]. Many of these discrete formulations can be transferred to the continuous case as shown in [3, 34, 40]. This also holds true for our approach, for which the discretized version of the dynamics (Eq. (3), (4)) can be easily cast into one of the existing frameworks (e.g [39]). These works have also been extended to the probabilistic context using a variational auto-encoder [2, 35]. The hidden variables are thereby used to either encode the initial latent states or static information. In contrast to our work, none of these approaches allow for function uncertainty in the dynamics. Finally, [41] proposes a symbolic physics framework, differing from our method in its search-based fixed grammar describing the dynamics.

Finally, we provide a summary of related techniques and derived comparison partners for our experiments in Table 1.

5 Experiments

We compare our approach against state-of-the-art methods in a large number of scenarios that differ in function complexity, signal-to-noise ratio and observability of the system. Our empirical findings suggest that our model leads to improved reliability of long-term predictions while being able to

successfully encapsulate autonomous dynamics from interactions effects. In all our experiments, we use the RK4 ODE solver and ACA library [42] for integration and gradient computation. Due to the space limit, we refer to the Supplementary Material for more detailed information about the experimental setup and comparison methods (see also Table 1 for an overview). Our PyTorch [43] implementation can be found in https://github.com/boschresearch/iGPODE (GNU AGPL v.3.0 license).

5.1 Experiment Details

Datasets We illustrate the performance of our model on two benchmark datasets: *bouncing balls* [44] and *charges* [2]. These datasets involve freely moving $N_\text{a} = 3$ balls that collide with each other and $N_\text{a} = 5$ particles that carry randomly assigned positive or negative charges leading to attraction or repulsion, respectively. All simulations are performed in frictionless, square boxes in $2D$. We generate 100 bouncing balls training sequences with different levels of Gaussian perturbations to simulate measurement noise (see Supplementary Section A.3 for details). Since the charges dataset requires inferring the charge information, we use 10k train sequences without observation noise as in [2] and similarly use 500 training sequences when velocity information is missing.

Partial observability Prior works typically evaluate their methods on datasets in which the observations consist of position and velocity [1, 2]. However, having access to the full state of an object is for many real-world problems unrealistic, e.g. many tracking devices can measure positions, but cannot measure velocity or acceleration. To test the model performance in such scenarios, we consider an additional bouncing balls dataset in which the velocities are not observed. On the charges dataset, we assume that position and velocity are observed, but treat the charge information as missing.

Reported metrics We quantify the model fit by computing the expected log-likelihood (ELL) of the data points under the predictive distribution. Further, we report the mean squared error (MSE) between ground truth and predictions over all predicted time steps and over all objects (see Supplementary Section A.3 for the exact definitions). Each experiment is repeated five times and we report the mean and standard deviation of both metrics on test sequences.

5.2 Empirical Findings

Due to our latent variable construction, ODE state dimensionality D could be arbitrary even though the observations are four-dimensional. To choose an appropriate state dimensionality, we study if the model performance can be improved by augmenting the state space with auxiliary dimensions ($D > 4$). Table 8 shows that increasing the model flexibility beyond need leads to overfitting as we observe lower training but significantly higher test error. Consequently, if not stated otherwise, we use a four-dimensional latent space for each object that corresponds to position and velocity in x and y coordinates and observations consist of their noisy versions. Next, we discuss the main findings.

5.2.1 Interacting dynamics are superior over standard formulation

We consider three *bouncing ball* datasets with varying noise levels to reflect different levels of problem difficulties. To demonstrate the merits of our decomposed formulation in Eq. (4), we compare it against a standard, GP-based non-interacting dynamical model (GPODE) in which the time differential takes the whole state vector $\mathbf{H}(t)$ as input [11]. As shown in Table 2, our interaction model consistently outperforms its standard counterpart irrespective of the noise level and function approximator. We also note that the results are consistent when we replace the GP approximation with deterministic and Bayesian neural networks (NNs), which indicates the robustness of the inductive bias. (see Table 12).

Table 2: Comparison of standard and interacting GP-ODE systems on bouncing ball datasets.

Noise Level	Model	MSE ↓	ELL ↑
No Noise	I-GPODE	$\mathbf{17.3 \pm 1.2}$	$\mathbf{-25.9 \pm 3.6}$
	GPODE	23.9 ± 0.9	-597.6 ± 28.5
Low Noise	I-GPODE	$\mathbf{17.8 \pm 0.9}$	$\mathbf{-26.5 \pm 4.5}$
	GPODE	23.6 ± 0.7	-580.8 ± 19.8
High Noise	I-GPODE	$\mathbf{18.4 \pm 0.9}$	$\mathbf{-29.4 \pm 1.1}$
	GPODE	24.5 ± 1.0	-569.3 ± 27.5

5.2.2 GP approximation yields more calibrated uncertainties

Next, we compare our GP interaction model (I-GPODE) against neural network variants with/without uncertainty (I-NODE, I-BNODE), obtained by replacing GPs with NNs and Bayesian NNs. The findings in Table 3 strongly indicate that I-GPODE learns better calibrated uncertainties, measured by ELL, than I-BNODE and I-NODE (see Figure 2 and 5 for visual illustrations). In terms of MSE, we observe that I-NODE tends to attain the smallest values. However, increasing noise levels deteriorates the performance of I-NODE and eventually the results become comparable to I-GPODE. The results stay consistent when we repeat the experiment with two balls (see Table 11).

Table 3: Comparison of our GP approximation against alternative function approximators.

Noise Level	Model	MSE ↓	ELL ↑
No Noise	I-GPODE	17.3 ± 1.2	$\mathbf{-25.9 \pm 3.6}$
	I-NODE	$\mathbf{8.5 \pm 0.9}$	-61.9 ± 8.6
	I-BNODE	20.4 ± 0.6	-37.8 ± 4.2
Low Noise	I-GPODE	17.8 ± 0.9	$\mathbf{-26.5 \pm 4.5}$
	I-NODE	$\mathbf{13.3 \pm 0.9}$	-96.2 ± 12.5
	I-BNODE	21.0 ± 0.8	-36.4 ± 2.8
High Noise	I-GPODE	18.4 ± 0.9	$\mathbf{-29.4 \pm 1.1}$
	I-NODE	$\mathbf{15.6 \pm 1.5}$	-128.9 ± 23.2
	I-BNODE	20.8 ± 0.7	-47.6 ± 10.5

5.2.3 GP approximation enables disentangled function learning

In the following, we study whether the estimated functions can disentangle independent kinematics from interaction effects. We train I-GPODE, I-NODE and I-BNODE on a dataset with three balls and evaluate the trained independent dynamics function $\mathbf{f}_s$ on a test dataset with one ball (since the test dataset incorporates a single object, the dynamics do not involve the interaction function $\mathbf{f}_b$). Three test sequences as well as the independent dynamics function predictions are illustrated in Figure 3 (see Table 9 for a quantitative comparison and Figure 4 for additional illustrations). We observe that I-NODE predictions tend to deviate from the test trajectory more quickly compared to I-GPODE predictions. We conjecture that this behaviour is because neural networks are overflexible and thus the learned functions may not necessarily decompose independent kinematics from interaction effects whereas the function-level regularization of I-GPODE helps with disentanglement.

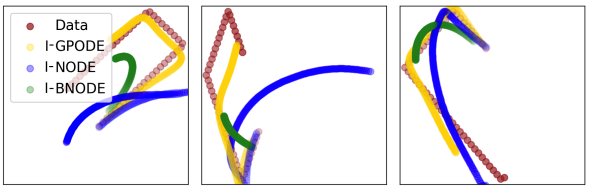

Figure 3: Disentanglement between independent kinematics and interactions. Our model variants are trained on a bouncing ball dataset with three balls and tested with a single ball using the independent dynamics function $\mathbf{f}_s$ only. Shown are the mean predictions with circles darkened as time lapses.

5.2.4 Structured dynamics improve latent dynamics learning

In our last bouncing ball experiment, we move to a setting in which the velocities are no longer observed. First, we keep the velocities as latent states and contrast two variants of our model, i.e. with structured latent space (I-GPODE-L-S) and with unstructured latent space (I-GPODE-L). As shown in Table 4, injecting strong prior knowledge helps in this challenging setting in which the first order model clearly fails (see also Table 13 for training results). Finally, we compare I-GPODE-L-S with I-GPODE, which drops the velocity component from the latent space (hence learning the dynamics in the data space). As demonstrated in Table 4, I-GPODE is clearly outper-

Table 4: Comparison of I-GPODE variants on a bouncing ball dataset without velocity observations. The suffices "-L" and "-S" stand for latent variable model and structured state space.

Noise	Model	MSE ↓	ELL ↑
No Noise	I-GPODE	24.1 ± 1.2	-300 ± 23
	I-GPODE-L	23.3 ± 0.8	-176 ± 301
	I-GPODE-L-S	$\mathbf{18.2 \pm 1.0}$	$\mathbf{-33 \pm 2}$
Low Noise	I-GPODE	23.3 ± 0.8	-239 ± 11
	I-GPODE-L	47.0 ± 47.4	-761 ± 922
	I-GPODE-L-S	$\mathbf{20.5 \pm 0.9}$	$\mathbf{-76 \pm 22}$
High Noise	I-GPODE	23.8 ± 0.7	-254 ± 16
	I-GPODE-L	119.3 ± 94.3	-920 ± 982
	I-GPODE-L-S	$\mathbf{21.8 \pm 0.7}$	$\mathbf{-115 \pm 19}$

formed by I-GPODE-L-S. It can thus be suggested that our latent variable construction is necessary in presence of missing states.

5.2.5 Global latent variables boost performance

In the final part of the experiments, we consider a more challenging dataset of charged particles. Since the dynamics are modulated by unknown charges, we turn to our global latent variable formulation. In other words, our learning task becomes simultaneously inferring the functions $\mathbf{f}_s$ and $\mathbf{f}_b$, the initial latent states $\mathbf{H}_1$, as well as a latent variable $\mathbf{c}^a \in \mathbb{R}$ associated with each observed trajectory $\mathbf{y}^a_{1:N}$. To form the upper and lower performance bounds, we include two baselines in which the charges are either observed or completely dropped from the model description.The results are shown in Table 5. We notice that the structured state space formulation boosts the performance of I-GPODE. However, the effect is less pronounced compared to the previous setting in which dynamic information is missing. See Table 14 for more results with different global variable encoders.

Table 5: Comparison of interacting GPODE and NODE systems on the charges dataset (no noise).

	MODEL	MSE ↓	ELL ↑
WITHOUT GLOBAL LATENTS	I-GPODE	19.2 ± 0.9	-171 ± 7
	I-NODE	14.1 ± 0.8	-460 ± 36
WITH GLOBAL LATENTS	I-GPODE	15.4 ± 2.3	-172 ± 88
	I-NODE	9.9 ± 0.6	-282 ± 14
	I-GPODE-S	12.9 ± 2.1	$\mathbf{-177 \pm 60}$
	I-NODE-S	$\mathbf{9.7 \pm 0.2}$	-282 ± 8
GLOBALS OBSERVED	I-GPODE	10.7 ± 1.1	-97 ± 9
	I-NODE	7.5 ± 1.2	-148 ± 27

6 Discussion

We have presented the first uncertainty-aware model for continuous-time interacting dynamical systems. By embedding the dynamics into a latent space, our formulation yields a flexible model family that can be applied to a variety of different scenarios. In our experiments, we found that our approach leads to improved disentanglement and superior calibration of long-term predictions.

Exploring useful applications of our disentangled representation is also an interesting direction for future research. Accurate identification of independent kinematics and interaction effects could enable useful downstream functionalities. For instance, one can perform algorithmic recourse [45] by counterfactual interventions at the object or interaction level.

Modeling limitations The capacity of our GP is limited by the choice of the kernel, e.g., the RBF kernel assumes that the dynamics are stationary. While our model formulation can be combined with arbitrary kernel functions and it is possible to increase the kernel expressiveness, e.g by building composite kernels or coming up with hand-crafted features, these approaches are often time-consuming and lead to highly parameterized kernels that are difficult to learn.

Approximation errors Our posterior inference scheme is inaccurate due to our variational framework, approximation errors that accumulate in time during future prediction, and numerical errors caused by numerical integration of ideally continuous dynamics and its solution.

Broader impact In our work, we propose a methodological contribution which is blind to specific data distributions. Its potential and unforeseeable side-effects in fairness-sensitive or safety-critical applications need to be investigated in a dedicated study.

Acknowledgements

The Bosch group is carbon neutral. Administration, manufacturing and research actvities do not longer leave a carbon footprint. This also includes GPU clusters on which the experiments have been performed.

We would like to thank Jakob Lindinger and Michael Herman for discussions and proofreading.

References

[1] Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Advances in Neural Information Processing Systems*, 2017.

[2] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, 2018.

[3] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *International Conference on Machine Learning*, 2020.

[4] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. Graph neural networks for modelling traffic participant interaction. In *IEEE Intelligent Vehicles Symposium*, 2019.

[5] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, 2020.

[6] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *IEEE International Conference on Robotics and Automation*, 2018.

[7] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2017.

[8] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models. In *Advances in Neural Information Processing Systems*, 2005.

[9] Roger Frigola and Carl Edward Rasmussen. Integrated pre-processing for Bayesian nonlinear system identification with Gaussian processes. In *IEEE Conference on Decision and Control*, 2013.

[10] César Lincoln C. Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A. Barreto, and Neil D. Lawrence. Recurrent Gaussian processes. *arXiv preprint arXiv:1511.06644*, 2016.

[11] Pashupati Hegde, Çağatay Yıldız, Harri Lähdesmäki, Samuel Kaski, and Markus Heinonen. Bayesian inference of ODEs with Gaussian processes. *arXiv preprint arXiv:2106.10905*, 2021.

[12] Katharina Ensinger, Friedrich Solowjow, Sebastian Ziesche, Michael Tiemann, and Sebastian Trimpe. Structure-preserving Gaussian process dynamics. *arXiv preprint arXiv:2102.01606*, 2021.

[13] Arno Solin, Ella Tamir, and Prakhar Verma. Scalable inference in SDEs by direct matching of the Fokker–Planck–Kolmogorov equation. In *Advances in Neural Information Processing Systems*, 2021.

[14] James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In *International Conference on Machine Learning*, 2020.

[15] Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ODE models with Gaussian processes. In *International Conference on Machine Learning*, 2018.

[16] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

[17] Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, 2019.

[18] Patrick Kidger, James Morrill, James Foster, and Terry J. Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, 2020.

[19] Cagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, 2021.

[20] Yuan Yin, Vincent Le Guen, Jérémie Dona, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. In *International Conference on Learning Representations*, 2021.

[21] Manuel Haußmann, Sebastian Gerwinn, Andreas Look, Barbara Rakitsch, and Melih Kandemir. Learning partially known stochastic dynamics with empirical PAC Bayes. In *International Conference on Artificial Intelligence and Statistics*, 2021.

[22] Zhaozhi Qian, William R Zame, Mihaela van der Schaar, Lucas M Fleuren, and Paul Elbers. Integrating expert ODEs into neural ODEs: Pharmacology and disease progression. In *Advances in Neural Information Processing Systems*, 2021.

[23] Carl E. Rasmussen and Christopher K.I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[24] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 2005.

[25] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, 2006.

[26] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2016.

[27] Roger Frigola. *Bayesian time series learning with Gaussian processes*. PhD thesis, University of Cambridge, 2015.

[28] Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. ODE2VAE: Deep generative second order ODEs with Bayesian neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[29] Moritz Diehl and Sébastien Gros. *Numerical Optimal Control*. University of Freiburg, 2017.

[30] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 2017.

[31] James Hensman, Nicolo Fusi, and Neil Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, 2013.

[32] Hugh Salimbeni, Vincent Dutordoir, James Hensman, and Marc Deisenroth. Deep Gaussian processes with importance-weighted variational inference. In *International Conference on Machine Learning*, 2019.

[33] Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, 2014.

[34] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. In *AAAI Conference on Artificial Intelligence*, 2020.

[35] Zijie Huang, Yizhou Sun, and Wei Wang. Learning continuous system dynamics from irregularly-sampled partial observations. In *Advances in Neural Information Processing Systems*, 2020.

[36] Lea Duncker, Gergo Bohner, Julien Boussard, and Maneesh Sahani. Learning interpretable continuous-time models of latent stochastic dynamical systems. In *International Conference on Machine Learning*, pages 1726–1734. PMLR, 2019.

[37] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems*, 2019.

[38] Raj Dandekar, Karen Chung, Vaibhav Dixit, Mohamed Tarek, Aslan Garcia-Valadez, Krishna Vishal Vemula, and Chris Rackauckas. Bayesian neural ordinary differential equations. *arXiv preprint arXiv:2012.07244*, 2020.

[39] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017.

[40] Michael Poli, Stefano Massaroli, Clayton M Rabideau, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Continuous-depth neural models for dynamic graph prediction. *arXiv preprint arXiv:2106.11581*, 2021.

[41] Kai Xu, Akash Srivastava, Dan Gutfreund, Felix Sosa, Tomer Ullman, Josh Tenenbaum, and Charles Sutton. A bayesian-symbolic approach to reasoning and learning in intuitive physics. *Advances in Neural Information Processing Systems*, 34:2478–2490, 2021.

[42] Juntang Zhuang, Nicha Dvornek, Xiaoxiao Li, Sekhar Tatikonda, Xenophon Papademetris, and James Duncan. Adaptive checkpoint adjoint method for gradient estimation in neural ODE. In *International Conference on Machine Learning*, 2020.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

[44] Ilya Sutskever, Geoffrey Hinton, and Graham Taylor. The recurrent temporal restricted Boltzmann machine. In *Advances in Neural Information Processing Systems*, 2008.

[45] A.H. Karimi, J.v. Kügelgen, B. Schölkopf, and I Valera. Algorithmic recourse under imperfect causal knowledge. In *Advances in Neural Information Processing Systems*, 2020.

[46] Imre Risi Kondor. *Group theoretical methods in machine learning*. Columbia University, 2008.

[47] David Ginsbourger, Xavier Bay, Olivier Roustant, and Laurent Carraro. Argumentwise invariant kernels for the approximation of invariant functions. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, 2012.

[48] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[49] Mark Van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems*, 2017.

[50] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplements Sec. A.3 and the attached code
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the "Dynamics approximations and hyper-parameter selection" paragraph in the supplements
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the "Training details" paragraph in the supplements
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Sec. 5.1
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our source code is attached
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Nonlinear Message Accumulations

Our formulation accumulates the messages coming from neighbors by computing their sum (Eq. 4). Although the experimental findings indicate that our seemingly simple construction successfully learns the underlying dynamics, we introduce two straightforward extension of our framework. First, one can learn an additional non-linear function $\mathbf{f}_\text{c}$ that takes all the incoming messages as input and generates the time differential as output:

$$\begin{aligned}\frac{d}{dt}\mathbf{h}^{a_n}(t) &= \mathbf{f}_\text{c}\left(\mathbf{m}^{n1}(t), \ldots, \mathbf{m}^{nN}(t)\right)\\ \mathbf{m}^{nn}(t) &= \mathbf{f}_\text{s}\left(\mathbf{h}^{a_n}(t), \mathbf{c}^{a_n}\right)\\ \mathbf{m}^{nm}(t) &= \mathbf{f}_\text{b}\left(\mathbf{h}^{a_n}(t), \mathbf{h}^{a_n'}(t), \mathbf{c}^{a_n}, \mathbf{c}^{a_m}\right),\end{aligned}$$

For notational convenience, we drop the neighboring graph from our write-up. In practice, $\mathbf{f}_\text{c}$ would only receive the messages coming from the neighbors. Since the messages interact in a non-linear way, this construction no longer disentangles the independent kinematics from the interactions.

Next, we introduce another construction in which the neighboring messages are weighted via a nonlinear function $\mathbf{f}_w$:

$$\begin{aligned}\frac{d}{dt}\mathbf{h}^{a}(t) &= \tilde{w}_{aa}\mathbf{f}_\text{s}\left(\mathbf{h}^{a}(t), \mathbf{c}^{a}\right) + \sum_{a'\in\mathcal{N}_a} \tilde{w}_{aa'}\mathbf{f}_\text{b}\left(\mathbf{h}^{a}(t), \mathbf{h}^{a'}(t), \mathbf{c}^{a}, \mathbf{c}^{a'}\right)\\ w_{aa'} &= \frac{\exp(w_{aa'})}{\exp(w_{aa}) + \sum_{n\in\mathcal{N}_a}\exp(w_{an})} \in (0,1)\\ w_{aa'} &= \mathbf{f}_w\left(\mathbf{h}^{a}(t), \mathbf{h}^{a'}(t), \mathbf{c}^{a}, \mathbf{c}^{a'}\right)\end{aligned}$$

When tested on bouncing ball datasets, this model achieved lower training and slightly higher test error (indicating overfitting). We leave further analysis of this new construction as an interesting future work.

A.2 Induced Kernel on Interaction Term

Next, we study our interaction component under a kernel perspective. In our formulation, we achieved permutation invariance across neighboring objects by aggregating all in-coming messages via the sum function (that is order invariant), similarly to what is done in standard graph neural network architectures [1].

It is interesting to see that the permutation invariance of our model formulation can also be derived from a kernel perspective. Seminal work on invariant kernels has been done by [46] and [47] who showed that a kernel is invariant under a finite set of input transformations, e.g. permutations, if the kernel is invariant when transforming its arguments. In our work, the Gaussian process prior on $\mathbf{f}_\text{b}$ induces a Gaussian process prior on the interaction term with covariance,

$$k_\text{i}(\mathbf{h}^p, \mathbf{h}^r) = \sum_{p'\in N_p}\sum_{r'\in N_r} k_\text{b}((\mathbf{h}^p, \mathbf{h}^{p'}), (\mathbf{h}^r, \mathbf{h}^{r'})),$$

which enforces the invariance by summing over all input combinations. Using a double sum is a common strategy for creating invariant kernels (see [48, 49] for a more in-depth discussion).

A.3 Experiment Details

Interaction function parameterization In all our experiments, we assume a fully connected object graph and also parameterize the interaction function $\mathbf{f}_\text{b}$ with the difference between object positions instead of absolute positions ($\mathbf{f}_\text{b}$ also takes the velocities and the global latent variables as input). Since the interactions are typically expressed in terms of distance, injecting an inductive bias of this sort helps to increase the performance as validated by our experiments.

Datasets To generate our datasets, we use the official implementations provided in [44, 2]. To ensure the accuracy of all numerical simulations in the respective code, we reduce the simulation step size. The dataset specifics are given in Table 6. Note that the noise values are proportional to the lower and upper limits of the position and velocity observations.

Table 6: Dataset details. We use the symbols P for the number of sequences, T for the sequence length, Δt for the time difference between consecutive observations, $\mathbf{s}_{\text{max}}$ and $\mathbf{v}_{\text{max}}$ for the maximum position and velocity observation, $\mathbf{s}_{\text{min}}$ and $\mathbf{v}_{\text{min}}$ for the minimum position and velocity observations, $\sigma_{\mathbf{s}}$ and $\sigma_{\mathbf{v}}$ for the standard deviation of the noise added to position and velocity observations, and T_{enc} for the length of the sequence needed for encoding and thereafter forward predictions.

DATASET	P_{tr}	P_{val}	P_{test}	T	Δt	$\mathbf{s}_{\text{max}}$	$\mathbf{v}_{\text{max}}$	$\mathbf{s}_{\text{min}}$	$\mathbf{v}_{\text{min}}$	T_{enc}	$\sigma_{\mathbf{s}}$	$\sigma_{\mathbf{v}}$
Noise-free bouncing balls	100	100	100	100	0.5	3.82	0.49	-3.82	-0.49	5	0.00	0.00
Low-noise bouncing balls	100	100	100	100	0.5	3.82	0.48	-3.82	-0.49	5	0.15	0.02
High-noise bouncing balls	100	100	100	100	0.5	3.82	0.48	-3.82	-0.48	5	0.30	0.04
Charges	10000	100	100	100	0.05	5.00	3.26	-5.00	-3.44	49	0.00	0.00

Dynamics approximations and hyper-parameter selection Our proposed I-GPODE method, in which the unknown independent kinematics and interaction functions are approximated with GPs, is compared with I-NODE and I-BNODE baselines in the bouncing balls experiment. To obtain the baselines, we simply replaced our GP approximation with multi-layer perceptrons (MLPs). We consider the standard, weight-space mean-field variational posterior for the BNN as done in [28]. In turn, the optimized dynamics parameters become the weights for I-NODE and the variational parameters for I-BNODE.

We perform an exhaustive comparison of hyper-parameters in different settings. In particular, for the independent kinematics function $\mathbf{f}_{\text{s}}$, we consider MLPs with two hidden layers and $N = 64/128/256/512$ hidden neurons, and sparse GPs with $M = 100/250/500$ inducing points. For the interaction function $\mathbf{f}_{\text{b}}$, we test with $N = 128/256/512$ hidden neurons and $M = 100/250/500/1000$ inducing points. We furthermore search the best activation function among `elu/relu/softplus/tanh/swish/lip-swish` activations and consider a diagonal or full lower-diagonal approximation to the covariance matrix (of the variational posterior). We test all hyper-parameter configurations on three validation datasets with varying noise levels. We found out that the `softplus` activation and diagonal covariance approximation consistently minimize the reported metrics on validation datasets. Other hyperparameters used in our experiments are reported in Table 7. Note that the number of parameters of the simpler, non-interacting models are approximately matched with the corresponding interacting model.

Table 7: The number of hidden neurons and inducing points used in our experiments.

MODEL	$\mathbf{f}_{\text{s}}$	$\mathbf{f}_{\text{b}}$
I-GPODE	$M = 250$	$M = 250$
I-NODE	$N = 256$	$N = 512$
I-BNODE	$N = 256$	$N = 512$
GPODE	$M = 250$	-
NODE	$N = 256$	-
BNODE	$N = 256$	-

Initial value encoder Inspired by previous work [17, 11], we infer the initial position and velocity variables using a RNN-based encoder architecture. Our encoder with GRU cells processes the first five observations in backward direction: $\mathbf{Y}_5 \rightarrow \mathbf{Y}_1$. The encoder output $\mathbf{z}_1 \in \mathbb{R}^{10}$ is mapped into

position and velocity initial value distributions via two separate MLPs that take the non-overlapping 5-dimensional chunks of $\mathbf{z}_1$ as input. Each MLP has the same architecture (one hidden layer, 50 neurons, ReLU activations). The model performance is somewhat robust against these encoder hyperparameters as validated by further comparisons. We finally note that the same encoder architecture is used for GP, NN and BNN-based models.

Latent variable encoder To infer the latent variables in the charges experiment, we again utilize an RNN-based encoder. Similar to the encoder used in [2], our architecture takes the first 49 observations as input, i.e., the global latent variable $\mathbf{c}^a$ associated with object a is extracted from all available observations $\mathbf{y}_{1:49}^{1:A}$. Since the overall performance crucially depends on the hyperparameter choices unlike the initial value extraction task, we consider two sets of encoders: a "large" encoder with $\mathbf{z}_1 \in \mathbb{R}^{100}$ and an MLP with 100 neurons as well as a "small" encoder with $\mathbf{z}_1 \in \mathbb{R}^{25}$ and an MLP with 50 neurons. We furthermore perform comparisons with `relu` and `elu` activation functions for the MLP. The results in the main paper are obtained with the "small" encoder with `elu` activation, which yields the best or runner-up performance across all settings.

Training details All model variants are trained with the Adam optimizer [50] with learning rates 5e-4, 5e-4 and 1e-4 for GP, NN and BNN-based models. We perform an incremental optimization scheme with three rounds, where randomly chosen 100 subsequences of length 5, 16, and 33 are used for training. We perform 25000, 12500 and 12500 optimization iterations in each round. Training each model respectively takes 9, 3 and 12 hours on NVIDIA Tesla V100 32GB. Finally, as proposed in [28], we stabilize the BNN learning by weighting the KL term $\mathrm{KL}[q(\mathcal{W})||p(\mathcal{W})]$ resulting from the BNN with a constant factor $\beta = D/|\mathcal{W}|$ in order to counter-balance the penalties on latent variables $\mathbf{h}^a \in \mathbb{R}^D$ and neural network weights $\mathcal{W} \in \mathbb{R}^{|\mathcal{W}|}$.

Reported metrics For a single trajectory $\mathbf{y}_{1:N}^{1:A}$, the MSE and ELL metrics are computed as follows:

$$\mathrm{MSE} = \frac{1}{LNA} \sum_{l=1}^{L} \sum_{n=1}^{N} \sum_{a=1}^{A} \left(\mathbf{y}_n^a - \hat{\mathbf{y}}_n^{a^{(l)}} \right)^2$$

$$\mathrm{ELL} = \frac{1}{NA} \sum_{n=1}^{N} \sum_{a=1}^{A} \log \mathcal{N}(\mathbf{y}_n^a; \hat{\boldsymbol{\mu}}_n^a, \hat{\boldsymbol{\sigma}}_n^a),$$

with $\hat{\mathbf{y}}_n^{a^{(l)}}$ being the l-th Monte Carlo prediction of object a at time point t_n and

$$\hat{\boldsymbol{\mu}}_n^a = \mathtt{mean}\left(\{\hat{\mathbf{y}}_n^{a^{(l)}}\}_{l=1}^L \right), \qquad \hat{\boldsymbol{\sigma}}_n^a = \mathtt{var}\left(\{\hat{\mathbf{y}}_n^{a^{(l)}}\}_{l=1}^L \right).$$

Finally, we report averages of the test statistics over all trajectories.

A.4 Additional Results

Latent neural ODE comparisons The ODE systems in our framework as well as the baseline models may be composed of positions, velocities and global latent variables. On the other hand, alternative black-box approaches [17, 28] typically consider a latent ODE system with arbitrary dimensionality and a VAE embedding between the observed and latent space. We compare these two modeling paradigms on the bouncing ball dataset. Since the reference methods are based on neural ODEs, we only consider neural network approximations for the differential functions. The results are presented in Table 8. In agreement with other comparisons, interaction models outperform their simpler, non-interacting counterparts. Also, latent ODE models tend to reduce the training error and increase the test error, which is a strong indicator of overfitting.

Table 8: A comparison of neural ODE variants on bouncing balls dataset, where the true, unknown ODE system is defined in $4D$. We suffix the dimensionality of the latent variables per object at the end of model names.

Noise Level	Model	Training		Test	
		MSE ↓	ELL ↑	MSE ↓	ELL ↑
No Noise	NODE	3.67 ± 0.35	-67.29 ± 8.04	30.07 ± 1.63	-911.95 ± 50.17
	I-NODE	6.50 ± 1.71	-36.85 ± 11.80	8.45 ± 0.93	-61.90 ± 8.63
	NODE-8	0.76 ± 0.22	-11.86 ± 9.03	29.53 ± 3.06	-885.24 ± 65.46
	I-NODE-8	2.25 ± 0.65	-14.44 ± 10.35	9.37 ± 1.45	-159.48 ± 34.13
	NODE-16	1.04 ± 0.38	-14.52 ± 6.88	26.39 ± 3.01	-615.33 ± 57.80
	I-NODE-16	1.64 ± 0.42	-4.01 ± 4.49	8.23 ± 0.81	-115.14 ± 30.98
Low Noise	NODE	2.35 ± 0.27	-34.82 ± 7.77	30.84 ± 1.94	-757.27 ± 56.12
	I-NODE	9.63 ± 0.79	-65.07 ± 11.77	13.33 ± 0.93	-96.15 ± 12.47
	NODE-8	0.76 ± 0.10	-9.12 ± 3.18	27.88 ± 1.93	-480.78 ± 46.87
	I-NODE-8	3.06 ± 1.17	-15.25 ± 4.56	15.29 ± 0.71	-167.43 ± 12.04
	NODE-16	1.46 ± 0.33	-18.71 ± 4.50	29.31 ± 2.76	-333.36 ± 30.79
	I-NODE-16	1.59 ± 0.41	-5.05 ± 2.06	15.40 ± 0.87	-155.66 ± 19.79
High Noise	NODE	2.83 ± 0.23	-43.38 ± 1.27	29.86 ± 1.38	-567.69 ± 47.49
	I-NODE	12.28 ± 1.09	-87.61 ± 13.27	15.60 ± 1.47	-128.89 ± 23.21
	NODE-8	1.14 ± 0.25	-24.70 ± 5.27	28.15 ± 1.48	-326.74 ± 32.69
	I-NODE-8	2.67 ± 0.40	-20.16 ± 2.65	20.76 ± 1.22	-154.67 ± 15.24
	NODE-16	1.94 ± 0.24	-28.39 ± 3.70	28.85 ± 3.14	-211.33 ± 21.87
	I-NODE-16	1.20 ± 0.09	-15.03 ± 1.84	20.74 ± 1.47	-146.14 ± 3.18
Higher Noise	NODE	4.02 ± 0.78	-62.19 ± 6.82	31.57 ± 0.56	-519.68 ± 37.86
	I-NODE	16.15 ± 0.21	-131.88 ± 10.11	18.35 ± 0.47	-164.60 ± 10.16
	NODE-8	1.55 ± 0.07	-44.16 ± 1.70	28.87 ± 1.09	-246.94 ± 19.09
	I-NODE-8	2.94 ± 0.42	-37.73 ± 2.34	23.54 ± 0.91	-119.53 ± 17.78
	NODE-16	2.90 ± 0.63	-44.17 ± 4.19	31.06 ± 1.21	-146.58 ± 19.16
	I-NODE-16	1.61 ± 0.22	-36.61 ± 3.79	24.67 ± 0.87	-143.61 ± 18.83

Table 9: Quantitative findings in the disentanglement task. We consider test sequences with different lengths. Note that all test sequences contain a single ball and thus the interaction function $\mathbf{f}_{\text{b}}$ does not play any role.

Model	$T_{\text{test}} = 10$		$T_{\text{test}} = 33$		$T_{\text{test}} = 100$	
	MSE ↓	ELL ↑	MSE ↓	ELL ↑	MSE ↓	ELL ↑
GPODE	0.21 ± 0.03	-2.37 ± 1.38	0.59 ± 0.17	-4.09 ± 3.30	9.84 ± 0.41	-17.58 ± 4.66
NODE	0.14 ± 0.05	-1.09 ± 2.51	2.90 ± 1.34	-130.33 ± 62.02	10.64 ± 2.87	-443.05 ± 145.56
BNODE	0.61 ± 0.05	-12.03 ± 1.60	2.39 ± 0.14	-23.11 ± 2.58	11.22 ± 0.54	-37.05 ± 2.95

Table 10: Discrete- vs continuous-time interacting GP models compared on a fully observed bouncing balls sequence with three balls.

Noise Level	Model	Training		Test	
		MSE ↓	ELL ↑	MSE ↓	ELL ↑
No Noise	I-GPODE	14.44 ± 0.69	-18.99 ± 2.12	17.29 ± 1.18	-25.88 ± 3.56
	I-GPSSM	14.80 ± 0.40	-17.16 ± 2.90	17.15 ± 1.05	-23.88 ± 4.09
Low Noise	I-GPODE	15.33 ± 0.46	-21.41 ± 4.41	17.76 ± 0.89	-26.48 ± 4.54
	I-GPSSM	14.99 ± 0.64	-17.28 ± 2.46	17.63 ± 0.79	-22.21 ± 3.51
High Noise	I-GPODE	16.10 ± 0.37	-24.05 ± 2.33	18.36 ± 0.86	-29.38 ± 1.14
	I-GPSSM	16.10 ± 0.72	-22.08 ± 2.28	18.45 ± 0.87	-24.86 ± 3.19
Higher Noise	I-GPODE	17.77 ± 1.36	-35.19 ± 1.56	20.06 ± 0.60	-41.76 ± 2.21
	I-GPSSM	17.35 ± 1.12	-33.67 ± 3.91	19.58 ± 0.45	-38.86 ± 1.68

Table 11: A comparison of NN and GP-based vanilla and interacting ODE systems on a bouncing balls dataset with two balls. All other settings are identical to the main experiment.

Noise Level	Model	Training		Test	
		MSE ↓	ELL ↑	MSE ↓	ELL ↑
No Noise	I-GPODE	10.18 ± 0.46	-6.38 ± 0.56	12.41 ± 0.90	-9.08 ± 0.71
	I-NODE	2.25 ± 0.28	-14.14 ± 4.19	4.53 ± 0.47	-56.24 ± 6.48
	I-BNODE	15.74 ± 0.63	-26.01 ± 2.23	16.33 ± 1.04	-24.18 ± 3.30
	GPODE	19.19 ± 0.72	-432.63 ± 8.55	19.68 ± 0.50	-442.38 ± 14.58
	NODE	2.99 ± 0.56	-47.59 ± 13.57	16.05 ± 0.57	-427.10 ± 32.06
	BNODE	16.73 ± 0.55	-27.50 ± 3.19	17.54 ± 0.74	-31.85 ± 1.48
Low Noise	I-GPODE	10.39 ± 0.27	-7.20 ± 0.99	13.10 ± 0.69	-10.74 ± 1.54
	I-NODE	4.39 ± 0.64	-33.49 ± 7.16	7.24 ± 0.36	-66.10 ± 3.30
	I-BNODE	16.35 ± 0.63	-25.76 ± 3.11	16.49 ± 0.51	-26.00 ± 2.96
	GPODE	20.05 ± 0.50	-450.15 ± 15.09	19.63 ± 0.55	-435.69 ± 9.70
	NODE	2.33 ± 0.53	-22.13 ± 0.17	16.17 ± 1.67	-303.51 ± 59.61
	BNODE	17.29 ± 0.52	-27.89 ± 2.60	17.88 ± 0.82	-30.06 ± 4.57
High Noise	I-GPODE	11.44 ± 0.46	-10.36 ± 1.03	13.20 ± 0.78	-13.63 ± 1.59
	I-NODE	6.13 ± 0.51	-51.98 ± 8.39	9.96 ± 0.82	-104.97 ± 9.98
	I-BNODE	15.81 ± 0.37	-27.55 ± 2.78	16.73 ± 0.59	-28.42 ± 4.55
	GPODE	19.91 ± 0.63	-423.58 ± 6.27	20.45 ± 0.31	-434.29 ± 13.16
	NODE	3.26 ± 0.64	-35.43 ± 4.23	16.99 ± 0.19	-232.94 ± 6.24
	BNODE	15.54 ± 0.22	-26.35 ± 2.56	17.01 ± 0.27	-26.08 ± 2.87
Higher Noise	I-GPODE	12.43 ± 0.55	-15.77 ± 1.66	14.83 ± 0.61	-20.28 ± 1.82
	I-NODE	8.18 ± 0.44	-72.30 ± 5.55	14.32 ± 0.95	-129.15 ± 19.09
	I-BNODE	16.16 ± 0.57	-29.08 ± 2.14	17.02 ± 0.73	-30.71 ± 3.46
	GPODE	19.68 ± 0.56	-392.48 ± 15.68	20.53 ± 0.46	-405.32 ± 12.33
	NODE	4.27 ± 0.80	-50.52 ± 2.81	18.91 ± 1.35	-228.16 ± 25.94
	BNODE	15.58 ± 0.40	-25.68 ± 2.47	16.98 ± 0.12	-27.82 ± 3.41

Table 12: A comparison of NN and GP-based vanilla and interacting ODE systems on a bouncing balls dataset with three balls. This experiment repeats the main findings plus the training results.

Noise Level	Model	Training		Test	
		MSE ↓	ELL ↑	MSE ↓	ELL ↑
No Noise	I-GPODE	14.44 ± 0.69	-18.99 ± 2.12	17.29 ± 1.18	-25.88 ± 3.56
	I-NODE	6.50 ± 1.71	-36.85 ± 11.80	8.45 ± 0.93	-61.90 ± 8.63
	I-BNODE	20.11 ± 0.77	-36.26 ± 4.63	20.39 ± 0.62	-37.81 ± 4.23
	GPODE	22.27 ± 0.60	-553.24 ± 13.25	23.89 ± 0.91	-597.63 ± 28.48
	NODE	3.67 ± 0.35	-67.29 ± 8.04	30.07 ± 1.63	-911.95 ± 50.17
	BNODE	20.89 ± 0.97	-47.16 ± 2.66	23.22 ± 0.60	-56.57 ± 9.41
Low Noise	I-GPODE	15.33 ± 0.46	-21.41 ± 4.41	17.76 ± 0.89	-26.48 ± 4.54
	I-NODE	9.63 ± 0.79	-65.07 ± 11.77	13.33 ± 0.93	-96.15 ± 12.47
	I-BNODE	20.28 ± 1.00	-38.31 ± 7.73	20.98 ± 0.78	-36.40 ± 2.77
	GPODE	23.07 ± 0.45	-564.22 ± 16.33	23.63 ± 0.72	-580.82 ± 19.83
	NODE	2.35 ± 0.27	-34.82 ± 7.77	30.84 ± 1.94	-757.27 ± 56.12
	BNODE	20.46 ± 0.62	-45.61 ± 6.46	22.99 ± 0.77	-51.98 ± 5.68
High Noise	I-GPODE	16.10 ± 0.37	-24.05 ± 2.33	18.36 ± 0.86	-29.38 ± 1.14
	I-NODE	12.28 ± 1.09	-87.61 ± 13.27	15.60 ± 1.47	-128.89 ± 23.21
	I-BNODE	20.31 ± 0.55	-38.04 ± 2.94	20.84 ± 0.72	-47.63 ± 10.51
	GPODE	22.75 ± 1.30	-532.03 ± 21.28	24.25 ± 1.00	-569.29 ± 27.46
	NODE	2.83 ± 0.23	-43.38 ± 1.27	29.86 ± 1.38	-567.69 ± 47.49
	BNODE	19.46 ± 0.71	-43.96 ± 1.45	22.33 ± 0.52	-52.49 ± 7.41
Higher Noise	I-GPODE	17.77 ± 1.36	-35.19 ± 1.56	20.06 ± 0.60	-41.76 ± 2.21
	I-NODE	16.15 ± 0.21	-131.88 ± 10.11	18.35 ± 0.47	-164.60 ± 10.16
	I-BNODE	20.34 ± 0.52	-52.49 ± 3.60	20.88 ± 0.55	-56.44 ± 5.71
	GPODE	23.85 ± 0.88	-517.26 ± 20.19	24.71 ± 1.25	-546.95 ± 31.79
	NODE	4.02 ± 0.78	-62.19 ± 6.82	31.57 ± 0.56	-519.68 ± 37.86
	BNODE	20.07 ± 0.98	-46.07 ± 3.54	22.36 ± 0.71	-54.53 ± 2.72

Table 13: A comparison of the standard GPODE against the latent I-GPODE variants on a bouncing ball dataset without velocity observations. Note that the suffix "-S" stands for structured state space. This table repeats the findings in the main paper, plus with training results.

Noise Level	Model	Training		Test	
		MSE ↓	ELL ↑	MSE ↓	ELL ↑
No Noise	I-GPODE	23.44 ± 0.39	-299.14 ± 21.03	24.11 ± 1.18	-300.78 ± 23.35
	I-GPODE-L	22.89 ± 0.44	-167.95 ± 25.76	23.32 ± 0.76	-176.04 ± 30.98
	I-GPODE-L-S	18.09 ± 0.27	-34.69 ± 2.45	18.22 ± 1.01	-33.45 ± 1.96
Low Noise	I-GPODE	23.50 ± 0.40	-243.65 ± 9.69	23.26 ± 0.77	-239.37 ± 11.10
	I-GPODE-L	49.26 ± 51.33	-810.91 ± 1025.94	47.01 ± 47.38	-760.60 ± 921.56
	I-GPODE-L-S	20.64 ± 0.21	-78.32 ± 22.79	20.46 ± 0.92	-76.35 ± 22.48
High Noise	I-GPODE	23.42 ± 0.89	-241.66 ± 17.19	23.77 ± 0.68	-254.34 ± 16.43
	I-GPODE-L	119.49 ± 96.91	-1306.78 ± 1277.32	119.24 ± 94.27	-1320.27 ± 1282.10
	I-GPODE-L-S	21.51 ± 0.45	-116.33 ± 11.57	21.80 ± 0.69	-115.50 ± 19.20
Higher Noise	I-GPODE	24.06 ± 0.38	-271.81 ± 9.72	24.57 ± 1.27	-267.08 ± 26.09
	I-GPODE-L	78.00 ± 29.98	-448.65 ± 299.58	78.49 ± 30.54	-450.42 ± 301.84
	I-GPODE-L-S	23.27 ± 0.49	-141.95 ± 16.96	23.64 ± 1.38	-147.35 ± 21.42

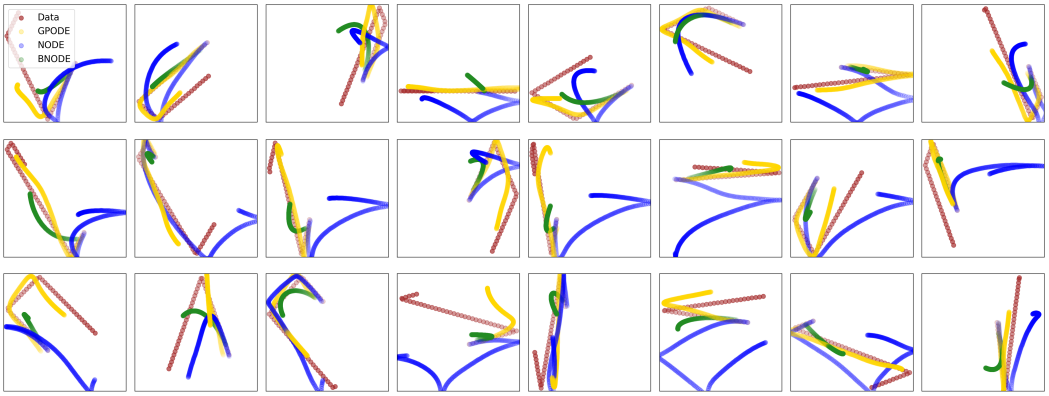

Figure 4: Additional plots comparing the independent kinematics functions of GP, NN and BNN based interacting ODE models. Above test trajectories contain single ball. Each row corresponds to one independent run of the experiment.

Table 14: A comparison of NN and GP-based interacting ODE systems on the charges dataset. Note that here we repeat the main findings plus the training results.

	Model	Training		Test	
		MSE ↓	ELL ↑	MSE ↓	ELL ↑
Without Global Latents	I-GPODE	19.3 ± 0.5	-170 ± 6	19.2 ± 0.9	-171 ± 7
	I-NODE	14.5 ± 0.7	-471 ± 23	14.1 ± 0.8	-460 ± 36
With Global Latents	I-GPODE	15.4 ± 1.8	-171 ± 85.7	15.4 ± 2.3	-172 ± 88
	I-NODE	9.8 ± 0.4	-271 ± 12.3	9.9 ± 0.6	-282 ± 14
	I-GPODE-S	12.8 ± 1.7	-176 ± 61	12.9 ± 2.1	-177 ± 60
	I-NODE-S	9.6 ± 0.3	-271 ± 12	9.7 ± 0.2	-282 ± 8
Globals Observed	I-GPODE	10.6 ± 0.6	-94 ± 8	10.7 ± 1.1	-97 ± 9
	I-NODE	7.3 ± 0.8	-140 ± 21	7.5 ± 1.2	-148 ± 27

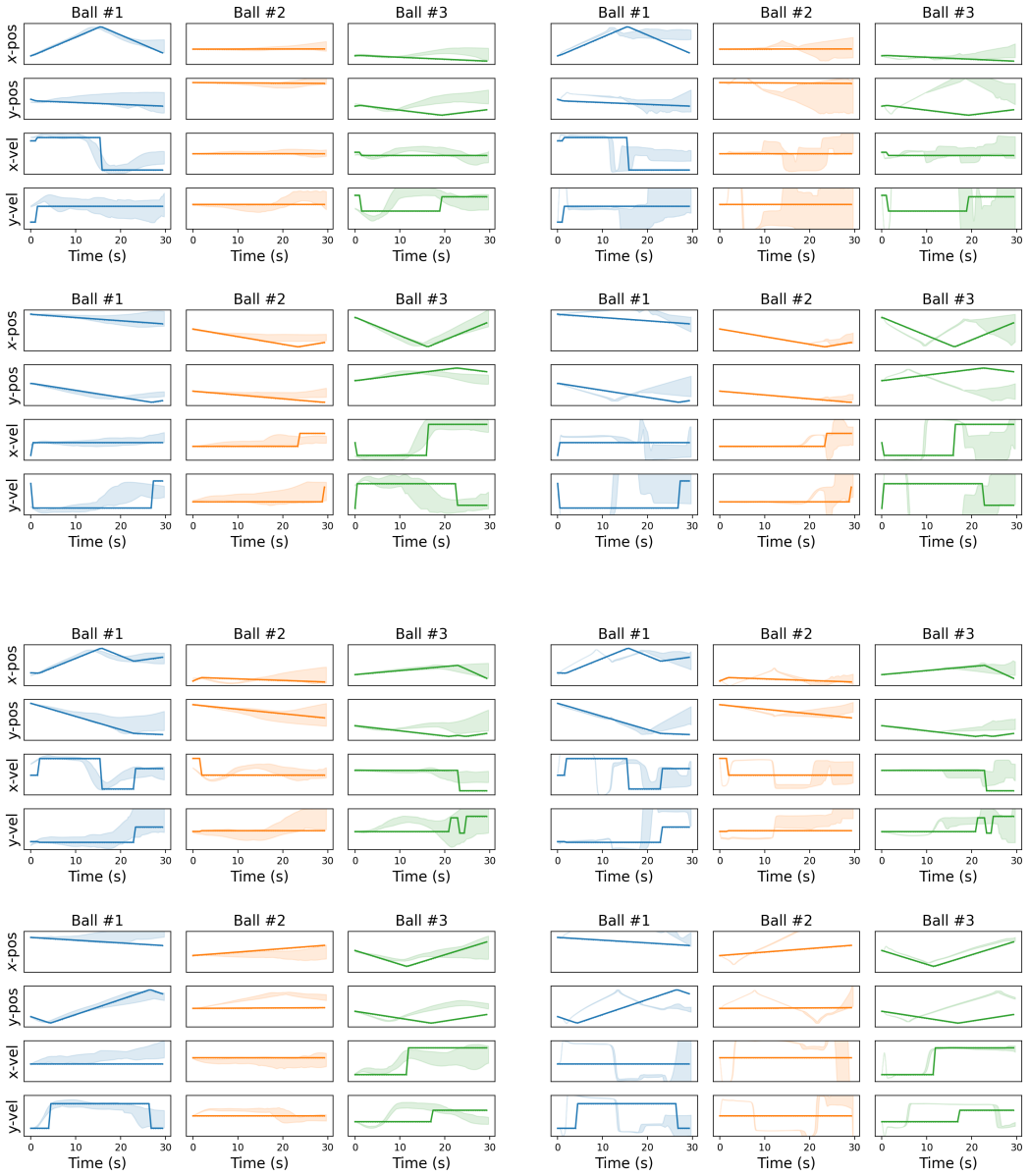

Figure 5: Additional plots comparing the uncertainty quantification of I-GPODE (left) and I-NODE (right) on bouncing balls dataset. The first two row groups show the training fits and the last two groups are the test predictions.