Foundation Inference Models for Ordinary Differential Equations

Maximilian Mauel [1 *] **Johannes R. Hübers** [2 3 *] **David Berghaus** [2 3] **Patrick Seifner** [1 2] **Ramsés J. Sánchez** [1 2 3]

Abstract

Ordinary differential equations (ODEs) are central to scientific modelling, but inferring their vector fields from noisy trajectories remains challenging. Current approaches such as symbolic regression, Gaussian process (GP) regression, and Neural ODEs often require complex training pipelines and substantial machine learning expertise, or they depend strongly on system-specific prior knowledge. We propose `FIM-ODE`, a pretrained Foundation Inference Model that *amortises* low-dimensional ODE inference by predicting the vector field directly from noisy trajectory data *in a single forward pass*. We pretrain `FIM-ODE` on a prior distribution over ODEs with low-degree polynomial vector fields and represent the target field with neural operators. `FIM-ODE` achieves strong zero-shot performance, matching and often improving upon `ODEFormer`, a recent pretrained symbolic baseline, across a range of regimes despite using a *simpler* pretraining prior distribution. Pretraining also provides a strong initialisation for *finetuning*, enabling fast and stable adaptation that outperforms modern neural and GP baselines *without requiring machine learning expertise*.

Our pretrained model, code repository, and tutorials will be released online shortly[1].

1. Introduction

The amortisation of inference procedures, through *pretraining* deep neural networks on large and heterogeneous *synthetic* datasets, is rapidly reshaping AI. This approach underlies many foundation models for time series forecasting (Dooley et al., 2024; Bhethanabhotla et al., 2024; Hemmer & Durstewitz, 2025) and imputation (Seifner et al., 2025b), prior fitted networks for tabular prediction (Müller et al., 2022; Hollmann et al., 2023; 2025), as well as models for causal discovery (Lorch et al., 2022; Kim et al., 2025), mutual information estimation (Gritsai et al., 2025), and dose response prediction in pharmacokinetics (Marin et al., 2025).

The idea is simple and (one could argue) stems from simulation-based inference (Cranmer et al., 2020). It consists in shifting the cost of inference from repeated dataset-specific optimization to a single, upfront pretraining phase across diverse synthetic datasets. Such pretraining pushes models to learn *reusable* inference algorithms that (ideally) do not depend on the specific conditioning context. The result is a class of foundation models that enable fast *zero-shot inference* on unseen data.

This trend has recently reached *system identification* in the form of Foundation Inference Models (FIMs). These models infer finite- or infinite-dimensional parametrizations of dynamical systems directly from noisy trajectories *in a single forward pass*. Examples include FIMs for continuous-time Markov chains (Berghaus et al., 2024), stochastic differential equations (Seifner et al., 2025a), and point processes (Berghaus et al., 2025). In this work we focus on ordinary differential equations (ODEs).

ODEs have played a fundamental role in scientific modelling across almost every discipline. They emerged as a language for celestial mechanics (Newton, 1687; Bernoulli, 1712), and remain a default model class for dynamical phenomena. Classical examples include concentration dynamics in molecular reaction networks (Hoff, 1986) and population oscillations in biology (Lotka, 1925; Volterra, 1927). They also provide some of the simplest settings exhibiting chaotic behaviour, with atmospheric convection as a canonical case (Lorenz, 1963). A first step toward amortised ODE inference was made by d'Ascoli et al. (2024) with `ODEFormer`. This model was pretrained on a very large corpus of synthetic ODEs drawn from a *complex* prior distribution, where vector fields are compositions of polynomial, trigonometric, and rational functions. Their goal was to recover the *symbolic expression* of the underlying vector field from noisy ODE solutions. We take `ODEFormer` as our primary baseline, and revisit amortised ODE inference through a simpler lens.

Starting from the classical intuition that simple dynamical

[1]University of Bonn [2]Lamarr Institute For Machine Learning and Artificial Intelligence [3]Fraunhofer IAIS, Germany. Correspondence to: Ramsés J. Sánchez <sanchez@cs.uni-bonn.de>.

Preprint. February 9, 2026.

[1]https://fim4science.github.io/OpenFIM/intro.html

rules can generate complex patterns (Kadanoff, 1986; 1987; Wolfram & Gad-el Hak, 2003), we ask whether a model pretrained on a much simpler prior can still generalise to real-world systems. Moreover, vector field estimation is only constrained in regions visited by the observed trajectories. This suggests a *local* viewpoint. We therefore ask whether, instead of a symbolic representation that is global in nature, one can represent the vector field locally, and obtain better accuracy in data-rich regimes. With this, our contributions are:

(1) We introduce a pretraining prior distribution over ODEs in dimensions one to three, with polynomial vector fields of degree at most three. We show that a model pretrained on this prior can estimate out-of-distribution vector fields, including vector fields that model human-motion trajectories.

(2) We represent inferred vector fields with neural operators. We show that this *local* representation can match and outperform *global* symbolic representations across a range of settings.

(3) We show that this pretraining also serves as a strong initialization, enabling fast and stable finetuning when the target dynamics are far out of distribution.

2. Related Work

To infer an ODE from data is to infer its underlying vector field. Non-parametric vector field inference methods fall mainly into three families, namely symbolic regression, Gaussian process (GP) regression, and neural ODE approaches. The first family aims for *symbolic* vector fields. Genetic programming methods search over symbolic expressions, but typically require time derivative estimates. They therefore rely on clean and densely sampled trajectories (Gaucel et al., 2014; La Cava et al., 2016; Quade et al., 2016; Kronberger et al., 2019; Atkinson et al., 2019; Weilbach et al., 2021), or on complex variational surrogates (Qian et al., 2022). SINDy (Brunton et al., 2016), a prominent alternative (Delahunt & Kutz, 2022; Brunton et al., 2025), assumes that the vector field admits a sparse linear representation in a *predefined* library of basis functions. This makes it sensitive to the choice of library and thus to prior knowledge, in addition to still requiring high-quality derivative information. The second family represents the vector field with Gaussian processes, which makes performance strongly dependent on the choice of the GP prior. Existing approaches have relied on gradient matching (Äijö & Lähdesmäki, 2009), on adjoint-based formulations with maximum a posteriori estimation (Heinonen et al., 2018) and, more recently, on mean-field variational approximations (Hegde et al., 2022). The third family corresponds to Neural ODEs (Chen et al., 2018; Yildiz et al., 2019; Rubanova et al., 2019; Dandekar et al., 2020), which avoids explicit priors by learning the vector field with neural networks. This flexibility comes at the cost of expensive and unstable training, due to backpropagation through numerical solvers or reliance on slow adjoint methods (Dupont et al., 2019; Finlay et al., 2020; Choromanski et al., 2020; Pal et al., 2021; Zhi et al., 2022). A parallel line of work resorts to neural variational inference, and parametrises both prior and posterior over vector fields with normalizing flows (Xu et al., 2025), but such methods are known to suffer from convergence issues (Adam et al., 2021; Verma et al., 2024). Finally, there is emerging work that leverages LLM agents to model vector fields using programs (Holt et al., 2024).

Overall, these approaches follow *the classical inference paradigm*, in which a model is optimised for a single dataset at a time. They rely on derivative estimation, careful prior design, solver-based training, or delicate variational optimisation, which can lead to complex training pipelines and, therefore, require substantial ML expertise. In contrast, amortised approaches learn an inference procedure once, through pretraining, and then apply it to new systems in a single forward pass. Besides `ODEFormer`, there are two other attempts at amortised ODE inference, one limited to one dimensional ODEs (Becker et al., 2023), and another that extends `ODEFormer` to contexts containing more than one trajectory (Şahin et al., 2025).

3. Preliminaries

In this section, we briefly introduce the ODE class we focus on, namely *autonomous, first-order* ordinary differential equations, and we formalise the data-driven ODE inference problem.

Ordinary differential equations. A d-dimensional, autonomous, first-order ODE is defined as

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}\left(\mathbf{x}(t)\right), \; t \in \mathbb{R}_+, \, \mathbf{x}(t) \in \mathbb{R}^d. \tag{1}$$

The vector-valued function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is the state-dependent *vector field*, and it fully characterizes the dynamics. Given some initial condition $\mathbf{x}(0)$ in $\mathbb{R}^d$, the ODE solution corresponds to a deterministic trajectory $\mathbf{x}(t)$, which we refer to as the *system state* over time. If the vector field $\mathbf{f}$ is locally Lipschitz in $\mathbf{x}$, the Picard-Lindelöf theorem guarantees that the corresponding initial value problem (IVP) has a unique solution in a neighbourhood of the initial condition (Arnold, 1992). This guarantee forms the foundation for both analytical and numerical approaches to solving ODEs. Appendix A provides additional background on existence and uniqueness results, numerical solution methods, and typical qualitative behaviour.

ODE inference problem. Consider a dynamical phenomenon whose evolution we observe through noisy and

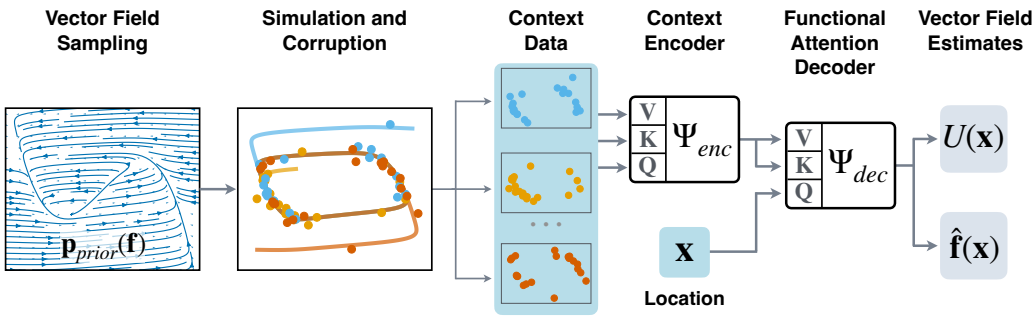


Figure 1. Synthetic data generation (left) and `FIM-ODE` architecture (right).

sparse measurements. Let $\mathcal{D}^* = \{(\mathbf{y}_1^*, \tau_1^*), \ldots, (\mathbf{y}_L^*, \tau_L^*)\}$ denote a dataset of L observations recorded at irregular time points $0 \leq \tau_1^* < \cdots < \tau_L^*$. We *assume* that each observation $\mathbf{y}_i^* \in \mathbb{R}^d$ corresponds to a noisy measurement of the (hidden) state $\mathbf{x}(\tau_i)$ of a dynamical system governed by the ODE above (Eq. 1), with an unknown vector field $\mathbf{f}^* : \mathbb{R}^d \to \mathbb{R}^d$. Our goal is to recover $\mathbf{f}^*$ from the observations alone.

More precisely, we seek to learn a parametric function $\hat{\mathbf{f}}_\theta : \mathbb{R}^d \times \mathcal{C} \to \mathbb{R}^d$, with parameters θ and $\mathcal{C}$ denoting the space of context datasets, such that when conditioning on $\mathcal{D}^*$, $\hat{\mathbf{f}}_\theta$ yields an accurate approximation of $\mathbf{f}^*(\mathbf{x})$ *in the region of state space visited by the observed trajectory*. Finally, we note that data-driven ODE inference is not only NP hard (Cubitt et al., 2012), but also subject to fundamental non-identifiability issues (Miao et al., 2011; Wang et al., 2024; Casolo et al., 2025). We nevertheless press on. In practice our aim is modest, namely to obtain a useful approximation of the vector field in the regions that are actually supported by trajectory data, and to *generalise* reliably within that regime.

Notation. We use $\mathbf{x}(t)$ to denote simulated ODE trajectories, and $\mathbf{x}^*(t)$ to denote hidden ODE processes assumed to underlie observed data. Similarly, $\mathbf{y}(t)$ refers to artificially corrupted trajectories, while $\mathbf{y}^*(t)$ denotes target data. Furthermore, we distinguish between the simulation discretization step Δt and the empirical inter-observation times $\Delta\tau$. Finally, we use the same symbols to denote probability distributions and their densities, as well as random variables and their realizations.

4. Foundation Inference Models For ODEs

We now introduce `FIM-ODE`, a pretrained model for *zero-shot* ODE inference from noisy trajectories. Our approach builds upon the Foundation Inference Model (FIM) framework, which amortises dynamical system inference by learning inference procedures during pretraining (Berghaus et al., 2024; 2025; Seifner et al., 2025b;a). The framework has two components. First, a pretraining prior distribution over the class of dynamical systems of interest. Second, a neural inference model that maps noisy simulated observations back to the parametrization of the underlying dynamics (*e.g.*, the vector field of an ODE). See Figure 1 for an illustration.

4.1. Pretraining Prior Distribution over ODEs

Our pretraining distribution is defined by three components, namely a prior over vector fields $p(\mathbf{f})$, a prior over initial conditions $p(\mathbf{x}_0)$, and a corruption mechanism. Accordingly, the data generation pipeline has three stages: (i) sample a vector field $\mathbf{f} \sim p(\mathbf{f})$; (ii) simulate multiple trajectories from each ODE by numerically integrating them from initial conditions $\mathbf{x}_0 \sim p(\mathbf{x}_0)$; and (iii) corrupt the trajectories with noise and subsampling to mimic realistic observations.

Prior over vector fields. We consider vector fields whose components are *sparse multivariate low-degree polynomials with random coefficients*, motivated by three considerations. First, many canonical ODEs used to model dynamical phenomena, from the Lorenz system to biological oscillators, are low-degree polynomial systems (see, *e.g.*, the collection in ODEBench (d'Ascoli et al., 2024)). Second, despite their simple algebraic form, polynomial vector fields generate a wide range of behaviors, including fixed points, limit cycles, and chaotic attractors. This aligns with the classical intuition that simple rules generate complex patterns. Third, polynomials are locally Lipschitz, which guarantees existence and uniqueness of trajectories by the Picard–Lindelöf theorem. Each component $f_i : \mathbb{R}^d \to \mathbb{R}$ is constructed as a multivariate polynomial of total degree *at most 3*. We sample coefficients independently from $\mathcal{N}(0, 1)$ and introduce sparsity by randomly masking out both degrees and individual monomials, yielding polynomials with varying structure and interaction patterns. We generate systems in dimensions $d \in \{1, 2, 3\}$. The full generation procedure, including monomial selection and sparsity control, is given in Appendix B.

Gaussian-process view. This construction admits a convenient GP interpretation. Conditional on a fixed monomial mask, each f_i is a finite-dimensional Gaussian process. That

is, each f_i is a linear combination of deterministic monomial features with Gaussian weights, and its kernel is the inner product of the corresponding feature vectors (equivalently, a sum over the active monomials). Marginalizing over the random mask yields a mixture of such GPs rather than a single GP, which preserves the same basic second-order structure but induces heavier-tailed variability across sampled systems. In either case, the prior is *nonstationary*, meaning that the pointwise variance $\mathrm{Var}(f_i(x))$ increases with $||\mathbf{x}||$ and is dominated by the highest-degree terms (scaling like $||\mathbf{x}||^{2\alpha}$, with α the highest degree, up to direction-dependent constants). See, for example, Figure 5 in the Appendix. Practically, this has consequence for our data generation, which we discuss in our Limitations section (Section 6).

Trajectory simulation. For each sampled vector field, we draw K initial conditions from $\mathcal{N}(0, 1)$, and integrate the ODE forward in time. We define an observation window $[0, 10]$ with 200 equidistant points, so that $\Delta t = 0.05$. We then integrate using Euler's method with 20 steps per observation interval, giving an integration step size of 0.0025. Finally, we discard systems that produce divergent trajectories, defined as trajectories whose magnitude exceeds 10^2. This focuses training on bounded regimes, and avoids wasting capacity on numerical blow ups.

Trajectory corruption. Real measurements are neither perfectly accurate nor uniformly sampled. To expose the model to these conditions, we corrupt simulated trajectories before training. We follow the corruption scheme of `ODEformer` (d'Ascoli et al., 2024), that is, we use multiplicative Gaussian noise, which keeps the signal-to-noise ratio in a controlled range, together with random subsampling. Concretely, we perturb the state as $y_i = (1 + \epsilon)x_i$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and we remove observations using an independent Bernoulli mask with probability $\rho \in [0, 0.5]$. All K trajectories from the same system share the same noise scale σ, but use independently sampled masks. This reflects a common experimental situation: measurement precision is fixed by the experimental setup, while the available timestamps can vary across repeated trials. Full details of the corruption procedure appear in Appendix B.

4.2. `FIM-ODE`: a Transformer-based Neural Operator Model

We now introduce a model that learns a parametric map $\hat{\mathbf{f}}_\theta : \mathbb{R}^d \times \mathcal{C} \to \mathbb{R}^d$, where θ are trainable parameters and $\mathcal{C}$ denotes the space of context datasets. Conditioned on a context $\mathcal{D} = \{(\mathbf{y}_{1k}, \tau_{1k}), \ldots, (\mathbf{y}_{L_k,k}, \tau_{L_k,k})\}_{k=1}^K$ of K noisy trajectories, the model returns a *local estimate* of the vector field at a query location $\mathbf{x}$. The aim is to approximate the vector field that best explains the observed trajectories, in the region of state-space visited by the data. This requires three capabilities. First, the model must process irregularly sampled, multi-trajectory observations. Second, it must relate query locations to nearby transitions in state space. And third, it must also generalise across systems with very different temporal and spatial scales. We address these requirements with a neural-operator architecture (Lu et al., 2021) built on attention mechanisms. The design follows an encoder-decoder structure. The encoder embeds the trajectory observations into a permutation-invariant context representation. The decoder then queries this representation at arbitrary spatial locations using cross attention, producing *local* vector field estimates.

Input normalization and scale invariance. Different ODEs come with different intrinsic scales. We promote *scale invariance* by normalising each state dimension to zero mean and unit variance, and by re-centring the distribution of inter-observation times $\Delta\tau$ around a target value. The model is trained in this normalised space. Predictions are then mapped back to the original coordinates using the chain rule. Full details appear in Appendix C.

Transition-based input representation. Rather than feeding raw trajectories, we construct transition features that encode local information. Each consecutive pair $(\mathbf{y}_i, \mathbf{y}_{i+1})$ defines a transition, from which we extract: (i) the current state $\mathbf{y}_i$; (ii) the displacement $\Delta\mathbf{y}_i = \mathbf{y}_{i+1} - \mathbf{y}_i$; (iii) the element-wise squared displacement $\Delta\mathbf{y}_i^2$; and (iv) the inter-observation times $\Delta\tau_i = \tau_{i+1} - \tau_i$. This is motivated by the structure of ODEs. The ratio $\Delta\mathbf{y}_i/\Delta\tau_i$ is a finite-difference estimate of the vector field at $\mathbf{y}_i$. The squared displacement provides a complementary second moment feature that captures the local scale of motion and the effective noise level under irregular sampling and measurement noise. Across the K trajectories of lengths $L_1, \ldots, L_k$, this extraction yields a set of $J = \sum_{k=1}^K (L_k - 1)$ transition tuples of the form $\tilde{\mathcal{D}} = \{(\mathbf{y}_i, \Delta\mathbf{y}_i, \Delta\mathbf{y}_i^2, \Delta\tau_i)\}_{i=1}^J$.

Context encoder. The encoder maps the context $\tilde{\mathcal{D}}$ into a permutation-invariant representation using self-attention. We first project each feature component independently to dimension $n/4$ using learnable linear projections (ϕ), then concatenate

$$\mathbf{d}_i = \mathrm{concat}\Big[\phi_{\mathbf{y}}(\mathbf{y}_i), \phi_{\Delta\mathbf{y}}(\Delta\mathbf{y}_i), \phi_{\Delta\mathbf{y}^2}(\Delta\mathbf{y}_i^2), \phi_{\Delta\tau}(\Delta\tau_i)\Big],$$

so that $\mathbf{d}_i \in \mathbb{R}^n$. Next, we apply two layers of linear self-attention (Katharopoulos et al., 2020) to the $n \times J$ matrix $\mathbf{D} = (\mathbf{d}_1, \ldots, \mathbf{d}_J)$, yielding a context representation $\mathbf{C} = \Psi_{enc}(\mathbf{D}, \mathbf{D}, \mathbf{D}) \in \mathbb{R}^{n \times J}$.

Functional attention decoder. Given a query location $\mathbf{x} \in \mathbb{R}^d$, the decoder extracts information from $\mathbf{C}$ via cross attention. We embed the location with a linear map $\phi_{\mathbf{x}}(\mathbf{x}) \in \mathbb{R}^n$, then pass it through M decoder blocks (ψ). Each block performs cross attention with queries from the location embedding and keys and values from $\mathbf{C}$ and returns $\mathbf{h}_i = \psi_i(\mathbf{h}_{i-1}, \mathbf{C}, \mathbf{C}) \in \mathbb{R}^n$, with $\mathbf{h}_0 = \phi_{\mathbf{x}}(\mathbf{x})$.

Table 1. ODEBench Trajectory Reconstruction. *Metric*: fraction of systems with variance weighted $R^2 > 0.9$. Higher is better.

Method	$\rho = 0.0$			$\rho = 0.5$		
	$\sigma = 0.0$	$\sigma = 0.03$	$\sigma = 0.05$	$\sigma = 0.0$	$\sigma = 0.03$	$\sigma = 0.05$
ODEformer	63.1%	61.5%	61.5%	63.9%	66.4%	61.5%
FIM-ODE	84.4%	80.3%	75.4%	82.8%	74.6%	72.1%

Table 2. ODEBench Trajectory Generalization. *Metric*: fraction of systems with variance weighted $R^2 > 0.9$. Higher is better.

Method	$\rho = 0.0$			$\rho = 0.5$		
	$\sigma = 0.0$	$\sigma = 0.03$	$\sigma = 0.05$	$\sigma = 0.0$	$\sigma = 0.03$	$\sigma = 0.05$
ODEformer	27.9%	27.9%	25.4%	31.1%	32.8%	27.0%
FIM-ODE	29.5%	32.8%	29.5%	27.0%	27.0%	28.7%

A final MLP maps the result to $\mathbb{R}^d$, so that $\hat{\mathbf{f}}_\theta(\mathbf{x}|\tilde{\mathcal{D}}) = \Psi_{dec}(\mathbf{h}_M(\mathbf{x}))$. This yields a continuous vector field estimator. We can evaluate $\hat{\mathbf{f}}_\theta$ at any point in state space, not only at observed states. This architecture is adapted from Seifner et al. (2025a).

Vector field loss with uncertainty weighting. We train by sampling query locations and matching predicted vector field values to ground truth. At each training step, we use a mixed sampling strategy. Half of the queries are drawn uniformly over the spatial extent of the observed data, and the other half are drawn from states along the simulated trajectories of the target ODE. The base loss is an MAE between $\hat{\mathbf{f}}_\theta(\mathbf{x}|\tilde{\mathcal{D}})$ and $\mathbf{f}(\mathbf{x})$.

A practical issue is that flow magnitudes vary widely across state space. Near the origin $\|\mathbf{f}(\mathbf{x})\|$ can be close to zero, while other regions may exhibit much larger speeds. Without correction, optimisation overemphasises high-magnitude regions and neglects accuracy near regions with vanishing speed. We therefore follow Seifner et al. (2025a) and use *uncertainty weighting*. Alongside $\hat{\mathbf{f}}_\theta(\mathbf{x}|\tilde{\mathcal{D}})$ and auxiliary head predicts $U_\theta(\mathbf{x}, \tilde{\mathcal{D}})$, interpreted as a log variance. The objective becomes

$$\mathcal{L}_\theta = \mathbb{E}_{(\mathbf{x},\tilde{\mathcal{D}},\mathbf{f})}\left[e^{-U_\theta(\mathbf{x},\tilde{\mathcal{D}})}\|\hat{\mathbf{f}}_\theta(\mathbf{x}|\tilde{\mathcal{D}}) - \mathbf{f}(\mathbf{x})\| + U_\theta(\mathbf{x},\tilde{\mathcal{D}})\right],$$

which corresponds to a Laplace likelihood with heteroscedastic scale. The first term down-weights uncertain regions, while the second prevents degenerate solutions. The expectation over $\tilde{\mathcal{D}}$ and $\mathbf{f}$ is taken with respect to the pretraining prior distribution. Query location $\mathbf{x}$ are sampled using the mixed strategy described above.

5. Experiments

This section summarises our experimental setup, including pretraining, datasets, evaluation metrics, and baselines.

Pretraining. We pretrain a single 13M parameter model, with 8M parameters for FIM-ODE, and 5M parameters for the auxiliary head that models U_θ. Pretraining uses a synthetic dataset of 600K polynomial ODE systems, split into 80K 1D, 210K 2D, and 310K 3D systems. During pretraining, the number of context trajectories per system is sampled uniformly between 1 and 9. For each trajectory, we sample between 100 and 200 noisy observations. Additional details on data generation, architecture, training hyperparameters, and ablations are provided in Appendices C and E.

Finally, let us remark that nothing in our architecture prevents training FIM-ODE on higher-dimensional systems. We restrict to three dimensions only because of our current data generation pipeline (see the limitations discussion in Section 6).

Baselines. We first compare against ODEformer (d'Ascoli et al., 2024), an 86M-parameter transformer that amortises ODE inference by pretraining on roughly 50M synthetic ODE systems. These are drawn from a complex prior, where vector fields are compositions of polynomial, trigonometric, and rational functions. We then compare against a set of state-of-the-art methods trained under the classical *per-dataset* paradigm. This includes neural approaches such as GP-DNF (Xu et al., 2025), Bayesian Neural ODE (BNeuralODE) (Dandekar et al., 2020), NeuralODE (Chen et al., 2018), ODE2VAE (Yildiz et al., 2019), and LatentSDE (Solin et al., 2021), as well as GP-based approaches such as GPODE (Hegde et al., 2022) and npODE (Heinonen et al., 2018). We evaluate across settings that target different challenges, including forecasting, imputation, and inference on real-world human motion trajectories.

5.1. Experiment 1: ODEBench

We begin by evaluating the zero-shot inference capability of FIM-ODE by comparing it to ODEformer. We use ODEBench, a benchmark introduced with ODEformer, which contains 61 autonomous ODE systems[2] (d'Ascoli et al., 2024). The benchmark includes vector fields composed of trigonometric, exponential, and rational functions, together with reference solutions evaluated on a fixed grid of 512 time points. Many systems in the benchmark are canonical in scientific modelling, including, *e.g.*, the Van der Pol oscillator and the Lorenz attractor.

We follow the experimental protocol of ODEBench and consider two corruption mechanisms: multiplicative noise $y_i = (1+\epsilon)x_i$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and random subsampling, where a fraction ρ of observations is removed. The task is to infer the hidden vector field from the corrupted trajectory.

We assess the models on two tasks. The first is *trajectory reconstruction*, which measures whether integrating the inferred vector field from the ground-truth initial condition

[2]We exclude two systems with dimension greater than three, since FIM-ODE is pretrained for dimensions at most three.

reproduces the clean reference trajectory on the fixed 512 point grid. The second is *trajectory generalisation*, which evaluates whether the inferred vector field produces accurate trajectories from new initial conditions not present in the context. We measure performance using the variance-weighted R^2 score, and report the fraction of test systems achieving $R^2 > 0.9$, following the ODEBench protocol (d'Ascoli et al., 2024). Tables 1 and 2 summarise performance on ODEBench across all corruption settings.

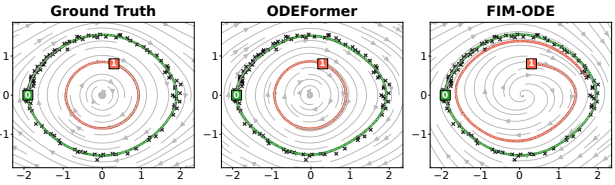
For trajectory reconstruction (Table 1), FIM-ODE consistently outperforms ODEformer across all noise and subsampling configurations. Performance degrades smoothly as corruption increases, suggesting that FIM-ODE has internalised the noise and subsampling mechanisms seen during pretraining. For trajectory generalisation (Table 2), the two methods perform comparably, although FIM-ODE shows a clearer advantage at less stringent thresholds (see Appendix E). These results are notable given that FIM-ODE is about *ten times smaller* (8M versus 86M parameters for the vector field predictor) and is pretrained on about *eighty times fewer systems* (0.6M versus 50M), highlighting the benefit of FIM-ODE 's local vector field representation.

A key point is that ODEBench contains substantial *out-of-distribution* (OOD) structure relative to our polynomial pretraining prior. Roughly one third of the systems have non-polynomial vector fields, including trigonometric and rational components. Despite this mismatch, Tables 1 and 2 show that FIM-ODE reconstructs trajectories accurately and can generalise competitively, which already hints at the value of a local estimator. Figure 2 makes the mechanism concrete on two non-polynomial systems.

First, consider the frictionless pendulum. Its vector field contains a sine term, and the phase portrait is organised by closed low-energy orbits near the origin. In this regime, FIM-ODE tends to bias the field toward a weak sink near the origin, a preference that is consistent with our polynomial prior. As a result, it can still match the context trajectory over the observed time window, yet it fails to preserve the correct conservative geometry, and therefore generalises poorly to unseen initial conditions. ODEformer, for which the sine functional form is within its prior, recovers the vector field more faithfully and preserves the expected closed orbits. Now, consider the reduced CDIMA model (same Figure), whose vector field contains rational terms. Here the relevant behaviour is largely controlled by local features in the region visited by the data. FIM-ODE captures these features well and performs strongly in both reconstruction and generalisation, while ODEformer is less accurate *despite being in-distribution*.

These examples illustrate why a local representation can be advantageous. FIM-ODE is only asked to approximate the vector field where the data provide constraints, and *local estimates can transfer*, even when the global functional form is OOD. By contrast, ODEformer targets a *global symbolic expression*. This can be powerful when the correct functional family is within distribution, but it can also force global commitments in regions that are not supported by observations. Overall, these results demonstrate strong zero-shot ODE inference under realistic corruption, and they highlight a central trade-off between local and global representations. Additional results on synthetic polynomial systems, including OOD generalisation to higher degree polynomials, are provided in Appendix E.3.

(a) Frictionless Pendulum (ODE 28 in ODEBench)



(b) Reduced Model for Chlorine Dioxide-iodine-malonic Acid (CDIMA) Reaction (ODE 42 in ODEBench)

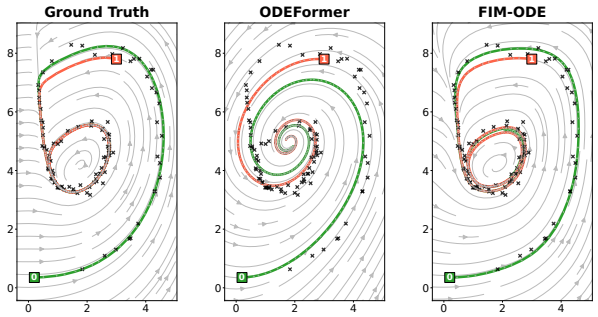


Figure 2. Comparison of ODEformer and FIM-ODE on two ODEBench systems. Each model infers a vector field from a single corrupted context trajectory, obtained by subsampling the ground truth ($\rho = 0.5$) and adding noise ($\sigma = 0.03$). The clean context trajectory is shown in green in the left column. We then integrate the inferred vector fields from two initial conditions specified by ODEBench to assess reconstruction and generalisation.

5.2. Experiment 2: Low-data OOD Regime.

We now study how FIM-ODE behaves when context trajectories are very short or sparse, and how it can be adapted to this regime through *finetuning*, which amounts to training it on the observed trajectories, similar to neural ODEs. We outline the details of this training method in appendix D. We follow the setup of Hegde et al. (2022) and its later reproduction by Xu et al. (2025), and consider two canonical oscillators: Van der Pol (VDP) and FitzHugh Nagumo (FHN). Both systems lie within our polynomial pretraining prior (See Table 3). The OOD aspect is *not* the functional form, but the amount of information available, namely the context windows are much shorter than those seen during pretraining.

Table 3. Oscillator systems used in Experiment 2.

Van der Pol (VDP)	FitzHugh Nagumo (FHN)
$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ -x_1 + \frac{1}{2}x_2(1 - x_1^2) \end{bmatrix}$	$\dot{\mathbf{x}} = \begin{bmatrix} 3\left(x_1 - \frac{x_1^3}{3} + x_2\right) \\ \frac{1}{3}\left(0.2 - 3x_1 - 0.2x_2\right) \end{bmatrix}$

Forecasting the VDP oscillator. We simulate the VDP system from the initial condition $\mathbf{x}(0) = (-1.5, 2.5)$ over $t \in [0, 14]$. We use the first half of the trajectory, $t \in [0, 7]$, as context and forecast the second half, $t \in [7, 14]$. From the context window we sample 50 observations corrupted by additive Gaussian noise with variance $\sigma^2 = 0.05$. We also sample 50 target, clean observations on the forecast window for evaluation. We consider two forecasting scenarios: *Task 1* uses uniformly spaced observation times, while *Task 2* uses irregular times drawn uniformly at random. We compare `FIM-ODE` to `ODEformer` and to the classical baselines listed above, which are trained from scratch on each dataset. We report our results in Table 4.

In this low-data regime, both pretrained models are typically suboptimal relative to classical methods. However, we find that performance depends strongly on the particular noise realisation. Table 4 reports results for the exact dataset and noise seed used by Hegde et al. (2022). To quantify sensitivity, we repeat the experiment over 100 independent trials with different noise seeds and report summary statistics in Appendix E. For some noise realisations, `FIM-ODE` performs competitively and *can outperform most baselines in zero-shot mode*. We report the best such instance as `FIM-ODE` (*Selected noise*) in Table 4. This variability indicates that, given the short context and noise level, the observations often do not constrain the dynamics enough for reliable zero-shot forecasting.

We then finetune `FIM-ODE` on the fixed dataset of Hegde et al. (2022) by minimising a mean absolute error between model predictions and observations, *à la* Neural ODE. This rapidly adapts the model to the low-data regime and substantially improves forecasting performance.

Imputation in the FHN oscillator. We simulate the FHN system over $t \in [0, 5]$ and sample 25 regularly-spaced observations corrupted by additive noise with variance $\sigma^2 = 0.025$, again following Hegde et al. (2022). To create a structured missing-data regime, we remove all observations whose states fall in the quadrant $x_1 > 0$ and $x_2 < 0$. We then evaluate how well the inferred dynamics recover trajectories that traverse this unobserved region. Table 4 contains our results.

Here again, the pretrained models are suboptimal, although `FIM-ODE` consistently outperforms `ODEformer` in zero-shot mode. As in the VDP case, performance exhibits sensitivity to the noise realisation, and we report additional trials and statistics in Appendix E. We also include the best seed result in Table 4. Finetuning `FIM-ODE` on the fixed dataset used by Hegde et al. (2022) improves performance substantially, and yields the best results among all methods considered.

Table 4. MSE on low-data out-of-distribution evaluations: VDP forecasting and FHN imputation. Lower is better. T1 and T2 denote Task 1 and Task 2. `FIM-ODE` is zero shot, `FIM-ODE` (*Selected noise*) uses a favourable noise realisation, and `FIM-ODE` (*Finetuned*) is finetuned on the context dataset.

Method	VDP (T1)	VDP (T2)	FHN
`BNeuralODE`	1.45	1.68	0.24
`NeuralODE`	0.29	0.55	0.18
`npODE`	0.16	2.08	0.08
`GPODE`	0.13	0.21	0.07
`ODE2VAE`	0.13	0.19	0.07
`LatentSDE`	0.10	0.15	0.05
`GP-DNF`	0.03	**0.04**	**0.04**
`ODEformer`	0.22	0.45	1.70
`FIM-ODE`	0.89	0.40	0.33
`FIM-ODE` (*Selected noise*)	**0.02**	0.13	**0.04**
`FIM-ODE` (*Finetuned*)	0.16	0.26	**0.04**

The results highlight a basic limitation of the low-data regime. With short context windows and moderate noise, the data often do not contain enough information to reliably pin down the dynamics. This manifests as sensitivity to noise realisations in zero-shot mode. Finetuning partially resolves this by injecting dataset-specific information into the model parameters, enabling rapid adaptation — even when the context alone is insufficient.

5.3. Experiment 3: Human Motion Capture

We next evaluate `FIM-ODE` on real-world trajectories from the CMU Motion Capture database (Carnegie Mellon University, 2003), using subjects 09, 35, and 39, performing walking and running. Each recording is a multivariate time series with 50 marker-based coordinates. Following the preprocessing protocol of Hegde et al. (2022) and Xu et al. (2025), we apply principal component analysis (PCA) and project each sequence onto a five-dimensional latent representation, while reporting prediction error in the original observation space via the inverse map.

We assess both short- and long-horizon forecasting on held-out test sequences. Short contexts (*i.e.*, training) data contain 50 to 100 observations, while long contexts data contain 100 to 250 observations. Both ranges lie within the distribution of context lengths seen during `FIM-ODE` pretraining.

A direct comparison to `ODEformer` is not meaningful here, since `ODEformer` can process only a single context trajectory. `FIM-ODE` can condition on multiple trajectories, but it is pretrained for state dimension at most three. We therefore

restrict FIM−ODE's latent representation to the first three principal components, and set the remaining components to zero, so that all methods use the same inverse PCA map back to the original measurement space. Table 5 reports test MSE across subjects and forecast horizons.

The results highlight an information bottleneck. For subject 09, the first three principal components carry most of the variance, about 94%, while the fourth and fifth together account for only 2.5%. In this regime, little information is lost by the three-dimensional projection, and FIM−ODE performs remarkably well. In fact, it *outperforms all classical non-parametric baselines in zero-shot mode* on short sequences, and remains competitive on long sequences.

For subjects 35 and 39 the situation changes. The first three components explain about 86.5% and 92% of the variance, while the fourth and fifth carry a non negligible fraction, 7.7% and 5.1%, respectively. This matters for two reasons. First, the discarded components carry real dynamical information about the motion. Any model restricted to three dimensions is asked to predict in an observation space where part of the relevant signal has been removed. Second, even within the retained components, the effective dynamics need not be well approximated by an autonomous ODE once higher-dimensional structure is projected away. In other words, the three-dimensional PCA coordinates can inherit *memory* from the neglected degrees of freedom (Mori, 1965; Zwanzig, 1961), reducing the information that an ODE model can exploit. Both effects place FIM−ODE at a disadvantage in this setting, and they help explain the degradation in zero-shot performance when facing motion patterns that deviate more strongly from the pretraining distribution.

That said, *finetuning* FIM−ODE on the available subject-specific data leads to rapid gains. After adaptation, FIM−ODE improves substantially across subjects and horizons, and it outperforms most baselines in essentially all cases. See Figure 10 in the Appendix for an illustration of the gains from finetuning FIM−ODE. This supports the broader picture suggested by the earlier experiments: pretraining provides a strong inference prior, but in regimes where the context carries limited information about the underlying dynamics, modest finetuning can inject the missing system-specific details.

Table 5. Test MSE metrics for dynamics prediction task on CMU MoCap dataset.

Method	Subject 09		Subject 35		Subject 39	
	short	long	short	long	short	long
BNeuralODE	25.50	21.32	23.09	20.86	53.34	39.66
NeuralODE	27.53	33.83	36.50	23.54	115.38	53.51
npODE	17.91	19.76	26.24	22.83	92.80	55.94
GP-ODE	9.11	8.38	10.11	11.66	26.72	21.17
ODE2VAE	9.05	8.14	9.25	10.08	25.25	21.06
LatentSDE	7.46	6.45	7.57	7.65	21.25	18.72
GP-DNF	7.03	6.04	**6.72**	**7.03**	19.43	**16.21**
FIM−ODE (3D)	**6.10**	7.32	655.40	48.94	295.29	243.87
FIM−ODE (*Finetuned*) (3D)	7.55	**5.35**	6.92	11.73	**15.56**	19.89

6. Conclusions

In this work, we extended the Foundation Inference Model framework to *amortised inference of ordinary differential equations* (ODEs) from noisy trajectory data. Our approach has two components. The first is a pretraining prior distribution over ODEs with *polynomial vector fields of degree at most three*. The second is FIM−ODE, a neural operator model pretrained on trajectories drawn from this prior. Empirically, we showed that FIM−ODE achieves strong zero-shot performance, matching and often outperforming ODEformer, despite being about ten times smaller and trained on about eighty times fewer systems drawn from a much simpler prior. This supports the benefit of a local estimator over global symbolic expressions. We also showed that FIM−ODE can be rapidly *finetuned* to challenging out-of-distribution regimes, including real-world human-motion trajectories.

Limitations. *The main limitation* lies in the current prior. Our polynomial construction induces a non-stationary scaling, in which typical vector field magnitudes grow with the distance from the origin. As a result, many sampled ODEs generate trajectories that quickly leave the region of interest, or blow up in finite time. This effect becomes more pronounced as dimension increases, creating a *curse of dimensionality* for generating broad, high-dimensional ODE priors that still yield numerically stable and physically plausible trajectories over the desired horizon. This is the main reason we restrict pretraining to dimensions at most three. *A second limitation* is architectural. The maximum state dimension is currently fixed by design, which forces us to choose *a priori* the dimensionality the model can process.

Future work. We plan to address these limitations in two directions. First, we will explore alternative priors, including stationary GP based constructions, with the aim of generating higher-dimensional systems that remain stable while still exhibiting rich dynamics. Second, we will replace the fixed-dimensional design with *axial attention mechanisms* (Ho et al., 2019), so that dimensionality is not hard-coded but inferred from the input. Finally, we will investigate how pretrained FIM−ODE weights can serve as a regulariser for equation discovery in high-dimensional settings, following recent progress in this direction (Hinz et al., 2025). We will apply this idea to data-driven discovery benchmarks such as those in Champion et al. (2019) and, more broadly, to dynamical models of evolving text representations (Cvejoski et al., 2023; 2022).

Acknowledgments

This research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence.

References

Adam, V., Chang, P., Khan, M. E., and Solin, A. Dual parameterization of sparse variational gaussian processes. *Advances in neural information processing systems*, 34: 11474–11486, 2021.

Äijö, T. and Lähdesmäki, H. Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics*, 25(22): 2937–2944, 2009.

Arnold, V. I. *Ordinary differential equations*. Springer Science & Business Media, 1992.

Atkinson, S., Subber, W., Wang, L., Khan, G., Hawi, P., and Ghanem, R. Data-driven discovery of free-form governing differential equations. *arXiv preprint arXiv:1910.05117*, 2019.

Becker, S., Klein, M., Neitz, A., Parascandolo, G., and Kilbertus, N. Predicting ordinary differential equations with transformers. In *International Conference on Machine Learning*, pp. 1978–2002. PMLR, 2023.

Berghaus, D., Cvejoski, K., Seifner, P., Ojeda, C., and Sanchez, R. J. Foundation inference models for markov jump processes. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=f4v7cmm5sC.

Berghaus, D., Seifner, P., Cvejoski, K., Ojeda, C., and Sánchez, R. J. In-context learning of temporal point processes with foundation inference models. *arXiv preprint arXiv:2509.24762*, 2025.

Bernoulli, J. *Extrait de la Réponse de M. Bernoulli à M. Herman, datée de Basle le 7. Octobre 1710.* Mémoires de l'Ac. Royale des Sciences, Boudot, Paris, 1710, 1712.

Bhethanabhotla, S. K., Swelam, O., Siems, J., Salinas, D., and Hutter, F. Mamba4cast: Efficient zero-shot time series forecasting with state space models. *arXiv preprint arXiv:2410.09385*, 2024.

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

Brunton, S. L., Zolman, N., Kutz, J. N., and Fasel, U. Machine learning for sparse nonlinear modeling and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 8, 2025.

Carnegie Mellon University. CMU graphics lab motion capture database. http://mocap.cs.cmu.edu/, 2003.

Casolo, C., Becker, S., and Kilbertus, N. Identifiability challenges in sparse linear ordinary differential equations. *arXiv preprint arXiv:2506.09816*, 2025.

Champion, K., Lusch, B., Kutz, J. N., and Brunton, S. L. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Choromanski, K. M., Davis, J. Q., Likhosherstov, V., Song, X., Slotine, J.-J., Varley, J., Lee, H., Weller, A., and Sindhwani, V. Ode to an ode. *Advances in Neural Information Processing Systems*, 33:3338–3350, 2020.

Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Cubitt, T. S., Eisert, J., and Wolf, M. M. Extracting dynamical equations from experimental data is np hard. *Physical review letters*, 108(12):120503, 2012.

Cvejoski, K., Sánchez, R. J., and Ojeda, C. The future is different: Large pre-trained language models fail in prediction tasks. *arXiv preprint arXiv:2211.00384*, 2022.

Cvejoski, K., Sánchez, R. J., and Ojeda, C. Neural dynamic focused topic model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 12719–12727, 2023.

Dandekar, R., Chung, K., Dixit, V., Tarek, M., Garcia-Valadez, A., Vemula, K. V., and Rackauckas, C. Bayesian neural ordinary differential equations. *arXiv preprint arXiv:2012.07244*, 2020.

d'Ascoli, S., Becker, S., Schwaller, P., Mathis, A., and Kilbertus, N. ODEFormer: Symbolic regression of dynamical systems with transformers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TzoHLiGVMo.

Delahunt, C. B. and Kutz, J. N. A toolkit for data-driven discovery of governing equations in high-noise regimes. *IEEE Access*, 10:31210–31234, 2022.

Dooley, S., Khurana, G. S., Mohapatra, C., Naidu, S. V., and White, C. Forecastpfn: Synthetically-trained zero-shot forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. *Advances in neural information processing systems*, 32, 2019.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. How to train your neural ode: the world of jacobian and kinetic regularization. In *International conference on machine learning*, pp. 3154–3164. PMLR, 2020.

Gaucel, S., Keijzer, M., Lutton, E., and Tonda, A. Learning dynamical systems using standard symbolic regression. In *European Conference on Genetic Programming*, pp. 25–36. Springer, 2014.

Gritsai, G., Richards, M., Méloux, M., Cho, K., and Peyrard, M. Mist: Mutual information estimation via supervised training. 2025.

Hegde, P., Yıldız, Ç., Lähdesmäki, H., Kaski, S., and Heinonen, M. Variational multiple shooting for Bayesian ODEs with Gaussian processes. In *Uncertainty in Artificial Intelligence*. PMLR, 2022.

Heinonen, M., Yildiz, C., Mannerström, H., Intosalmi, J., and Lähdesmäki, H. Learning unknown ode models with gaussian processes. In *International conference on machine learning*, pp. 1959–1968. PMLR, 2018.

Hemmer, C. J. and Durstewitz, D. True zero-shot inference of dynamical systems preserving long-term statistics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=RE97LT26w8.

Hinz, M., Mauel, M., Seifner, P., Berghaus, D., Cvejoski, K., and Sanchez, R. J. Towards fast coarse-graining and equation discovery with foundation inference models. *arXiv preprint arXiv:2510.12618*, 2025.

Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

Hoff, J. H. v. t. *Studies in Chemical Dynamics*. Amsterdam: F. Mueller and Co. London: Williams and Norgate, 1986.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Holt, S., Qian, Z., Liu, T., Weatherall, J., and van der Schaar, M. Data-driven discovery of dynamical systems in pharmacology using large language models. *Advances in Neural Information Processing Systems*, 37:96325–96366, 2024.

Kadanoff, L. P. On two levels. *Physics Today*, 39(9):7–9, 09 1986. ISSN 0031-9228.

Kadanoff, L. P. On complexity. *Physics Today*, 40(3):7–9, 03 1987. ISSN 0031-9228.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Kim, J.-H., Gibbs, C. S., Yun, S., Song, H. O., and Cho, K. Large-scale targeted cause discovery via learning from simulated data. *Transactions on Machine Learning Research*, 2025.

Kronberger, G., Kammerer, L., and Kommenda, M. Identification of dynamical systems using symbolic regression. In *International Conference on Computer Aided Systems Theory*, pp. 370–377. Springer, 2019.

La Cava, W., Danai, K., and Spector, L. Inference of compact nonlinear dynamic models by epigenetic local search. *Engineering Applications of Artificial Intelligence*, 55: 292–306, 2016.

Lorch, L., Sussex, S., Rothfuss, J., Krause, A., and Schölkopf, B. Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35:13104–13118, 2022.

Lorenz, E. N. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101v3.

Lotka, A. J. *Elements of physical biology*. Williams & Wilkins, 1925.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Marin, C. A. O., Huisinga, W., Kavwele, P., and Hartung, N. Amortized in-context mixed effect transformer models: A zero-shot approach for pharmacokinetics. *arXiv preprint arXiv:2508.15659*, 2025.

Miao, H., Xia, X., Perelson, A. S., and Wu, H. On identifiability of nonlinear ode models and applications in viral dynamics. *SIAM review*, 53(1):3–39, 2011.

Mori, H. Transport, collective motion, and brownian motion. *Progress of theoretical physics*, 33(3):423–455, 1965.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Newton, I. *Philosophiae Naturalis Principia Mathematica*. cf. especially Liber de Motu I Sects. II, III, VII. London, 1687.

Pal, A., Ma, Y., Shah, V., and Rackauckas, C. V. Opening the blackbox: Accelerating neural differential equations by regularizing internal solver heuristics. In *International Conference on Machine Learning*, pp. 8325–8335. PMLR, 2021.

Qian, Z., Kacprzyk, K., and van der Schaar, M. D-code: Discovering closed-form odes from observed trajectories. In *International Conference on Learning Representations*, 2022.

Quade, M., Abel, M., Shafi, K., Niven, R. K., and Noack, B. R. Prediction of dynamical systems by symbolic regression. *Phys. Rev. E*, 94:012214, 2016.

Rubanova, Y., Chen, R. T., and Duvenaud, D. K. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Şahin, Y. E., Kilbertus, N., and Becker, S. Predicting symbolic odes from multiple trajectories. *arXiv preprint arXiv:2510.23295*, 2025.

Seifner, P., Cvejoski, K., Berghaus, D., Ojeda, C., and Sanchez, R. J. In-context learning of stochastic differential equations with foundation inference models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Seifner, P., Cvejoski, K., Körner, A., and Sanchez, R. J. Zero-shot imputation with foundation inference models for dynamical systems. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Solin, A., Tamir, E., and Verma, P. Scalable inference in SDEs by direct matching of the Fokker–Planck–Kolmogorov equation. *Advances in Neural Information Processing Systems*, 34:7071–7084, 2021.

Teschl, G. *Ordinary differential equations and dynamical systems*. Graduate studies in mathematics. American Mathematical Society, Providence, RI, 9 2012. ISBN 978-0-8218-8328-0. URL https://www.mat.univie.ac.at/~gerald/ftp/book-ode/.

Verma, P., Adam, V., and Solin, A. Variational gaussian process diffusion processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 1909–1917. PMLR, 2024.

Volterra, V. *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*, volume 2. Societá anonima tipografica" Leonardo da Vinci", 1927.

Wang, Y., Huang, W., Gong, M., Geng, X., Liu, T., Zhang, K., and Tao, D. Identifiability and asymptotics in learning homogeneous linear ode systems from discrete observations. *Journal of Machine Learning Research*, 25(154): 1–50, 2024.

Weilbach, J., Gerwinn, S., Weilbach, C., and Kandemir, M. Inferring the structure of ordinary differential equations. *arXiv preprint arXiv:2107.07345*, 2021.

Wolfram, S. and Gad-el Hak, M. A new kind of science. *Appl. Mech. Rev.*, 56(2):B18–B19, 2003.

Xu, J., Du, S., Yang, J., Ding, X., Zeng, D., and Paisley, J. Bayesian Gaussian process ODEs via double normalizing flows. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2025.

Yildiz, C., Heinonen, M., and Lahdesmaki, H. Ode2vae: Deep generative second order odes with bayesian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhi, W., Lai, T., Ott, L., Bonilla, E. V., and Ramos, F. Learning efficient and robust ordinary differential equations via invertible neural networks. In *International Conference on Machine Learning*, pp. 27060–27074. PMLR, 2022.

Zwanzig, R. Memory effects in irreversible thermodynamics. *Physical Review*, 124(4):983, 1961.

A. ODE Background

This appendix provides additional theoretical background on ordinary differential equations, including existence and uniqueness theorems, numerical solution methods, and typical dynamical behaviors.

A.1. Existence and Uniqueness of Solutions

Peano Theorem — Existence of a Solution. If the flow field $\mathbf{f}$ is continuous in $\Omega \times [t_0, t_0 + T]$, Peano's theorem states that any adhering initial value problem (IVP) will have at least one solution trajectory. Additionally, the solution trajectories will be contained in an ϵ-ball around $\mathbf{x}_0$ (Teschl, 2012).

Picard-Lindelöf Theorem — Existence of a Unique Solution. The Picard-Lindelöf theorem goes one step further and constrains $\mathbf{f}$ to be locally Lipschitz continuous with respect to $\mathbf{x}$ with the Lipschitz constant $L \in \mathbb{R}_+$ independent of t (Teschl, 2012):

$$\|\mathbf{f}(\mathbf{x}_1, t) - \mathbf{f}(\mathbf{x}_2, t)\| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|. \tag{2}$$

Every IVP that uses an $\mathbf{f}$ upholding those constraints is guaranteed a unique solution trajectory in an ϵ-ball around $\mathbf{x}_0$ and $t \in [t_0, t_0 + T]$.

A.2. Numerical Solving of Initial Value Problems

In practical environments, analytical methods for solving IVPs typically are not available. Fortunately, IVPs can also be numerically approximated using iterative procedures. One such family of methods, called one-step methods (OSM), is defined as:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \int_t^{t+\Delta t} \mathbf{f}(\mathbf{x}(\tau), \tau)\, d\tau \tag{3}$$

or more generally

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \phi(\mathbf{x}(t), t, \Delta t) \tag{4}$$

where $\Delta t \cdot \phi(\mathbf{x}(t), t, \Delta t)$ can be understood as an approximation of $\int_t^{t+\Delta t} \mathbf{f}(\mathbf{x}(\tau), \tau)\, d\tau$.

Common OSM choices include the Euler method, the trapezoidal method, and the Runge-Kutta method. Euler replaces $\Delta t \cdot \phi$ by $\Delta t \cdot \mathbf{f}(\mathbf{x}(t), t)$, while the Runge-Kutta method needs more evaluations of $\mathbf{f}$ but achieves higher accuracy per step with local accuracy of $\mathcal{O}(\Delta t^4)$ compared to Euler's $\mathcal{O}(\Delta t^1)$.

A.3. Typical ODE Behaviors

ODEs frequently produce solution trajectories that evolve according to distinctly identifiable structures in state space. Typical ODE features shaping trajectories are:

Equilibrium Points. Points $\mathbf{x}_e$ where the system velocity is zero: $\mathbf{f}(\mathbf{x}_e) = 0$. Equilibrium points can be classified as:

- *Attractors (sinks)*: trajectories converge to the equilibrium over time.
- *Sources*: trajectories diverge from the equilibrium.
- *Saddle points*: some directions attract while others repel.

Limit Cycles. Trajectories caught in periodic orbits. A limit cycle is characterized by a non-constant trajectory $\mathbf{x}(t)$ that repeats after some time $T > 0$, satisfying $\mathbf{x}(t + T) = \mathbf{x}(t)$.

Chaotic Behavior. Chaotic ODEs describe systems whose solution trajectories exhibit extreme sensitivity to initial conditions. This means that two close initial conditions lead to trajectories that diverge rapidly over time, quantified by positive Lyapunov exponents. The Lyapunov exponent λ measures how quickly close trajectories diverge: if the separation increases as $\|\delta(t)\| \approx \|\delta(0)\|e^{\lambda t}$, then λ is the Lyapunov exponent. A positive Lyapunov exponent ($\lambda > 0$) indicates chaotic behavior.

B. Data Generation Details and Dataset Statistics

This appendix provides comprehensive details on the synthetic data generation process for training `FIM-ODE`.

B.1. Polynomial ODE Generation

The generation process targets polynomial ODEs where each component function f_i is a multivariate polynomial of maximal degree p:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_d) \\ f_2(x_1, x_2, \ldots, x_d) \\ \vdots \\ f_d(x_1, x_2, \ldots, x_d) \end{bmatrix}, \quad f_i : \mathbb{R}^d \to \mathbb{R} \tag{5}$$

Each component function f_i is defined as an independent multivariate polynomial. Let $\mathbf{X}_d = \{x_1, x_2, \ldots, x_d\}$ be the set of variables. A monomial of degree p is determined by a set $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_d\}$, where $\alpha_i \in \mathbb{N}$ and $\sum_{i=1}^{d} \alpha_i = p$. Each $\mathcal{A}$ defines a monomial term:

$$\mathrm{M}(\mathcal{A}, \mathbf{X}_d) = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_d^{\alpha_d} \tag{6}$$

A polynomial is expressed as a weighted sum over all monomials up to degree p:

$$f_i(\mathbf{X}_d) = \sum_{j=0}^{p} \sum_{\forall \mathcal{A} \in \boldsymbol{\mathcal{A}_j}} c_{\mathcal{A}} \cdot \mathrm{M}(\mathcal{A}, \mathbf{X}_d), \quad \boldsymbol{\mathcal{A}_j} = \left\{ \mathcal{A} = \{\alpha_1, \ldots, \alpha_d\} \mid \sum_{k=1}^{d} \alpha_k = j \right\} \tag{7}$$

To manage complexity and variability, binary indicator variables $m^{degree} \in \{0, 1\}$ and $m^{monomial} \in \{0, 1\}$ are used to exclude random degrees and monomials:

$$f_i(\mathbf{X}_d) = \sum_{j=0}^{p} m_j^{degree} \cdot \left(\sum_{\forall \mathcal{A} \in \boldsymbol{\mathcal{A}_j}} m_{\mathcal{A}}^{monomial} \cdot c_{\mathcal{A}} \cdot \mathrm{M}(\mathcal{A}, \mathbf{X}_d) \right) \tag{8}$$

After generating a polynomial for each dimension, a global scaling factor s is applied:

$$\frac{d\mathbf{x}(t)}{dt} = s \cdot \mathbf{f}(\mathbf{x}(t)) \tag{9}$$

Implementation Parameters. Coefficients $c_{\mathcal{A}}$ are sampled from $\mathcal{N}(0, 1)$. The masks m^{degree} and $m^{monomial}$ are sampled uniformly with constraints ensuring at least one degree and one monomial per polynomial are retained. The scaling factor s is drawn from a uniform distribution over $[0, 2]$.

B.2. Trajectory Generation and Filtering

Trajectories are generated by solving IVPs from selected initial conditions. For each ODE system, $K = 9$ distinct initial conditions are randomly sampled from $\mathcal{N}(0, 1)$. Trajectories are gathered by numerically integrating the ODE using Euler integration over the time interval $T = [0, 10]$ with $n_{points} = 200$, resulting in a temporal resolution of $\Delta t = 0.05$. The Euler integration uses $n_{intermediate} = 20$ intermediate steps between each point on the grid.

Data Filtering. ODEs are discarded if any of their trajectories diverge (any observation exceeds $\delta_{reject} = 10^2$ or becomes non-finite).

B.3. Dataset Construction and Corruption

The training dataset contains trajectories and matching samples from the vector fields of the ODE systems. To determine where to evaluate the vector field, a bounding box is drawn around the generated trajectories and expanded by a factor $s_{bbox} = 0.2$. Within this expanded bounding box, the ODE's vector field is sampled at $n_{vf} = 10000$ random locations.

Data Corruption. Training examples are corrupted using multiplicative Gaussian noise and subsampling. The noise scale σ is sampled from $[0, 0.06]$, and the subsampling rate ρ is sampled from $[0, 0.5]$. All trajectories within an ODE system are exposed to the same level of noise, but each trajectory has different points subsampled.

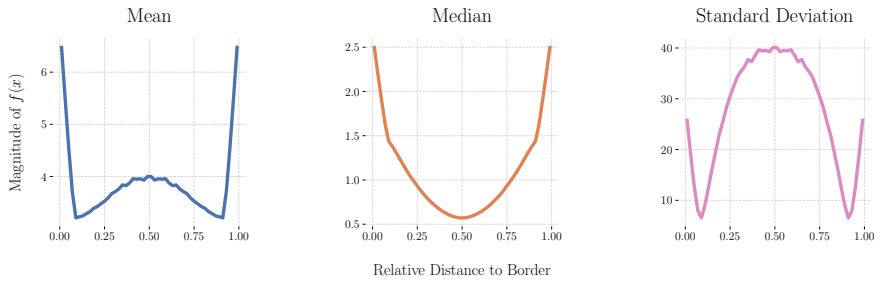


Figure 3. Magnitude statistics of vector field points as a function of relative distance to bounding box borders for 1D ODEs. The 1D case exhibits unique behavior due to outlier systems (see Figure 6).

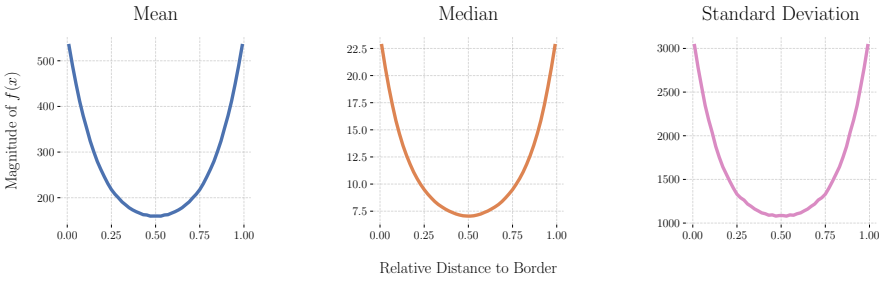


Figure 4. Magnitude statistics of vector field points as a function of relative distance to bounding box borders for 2D ODEs.

Dataset Sizes. The pretraining dataset contains 600,000 polynomial ODE systems spanning dimensionality 1, 2, and 3 (80,000 one-dimensional, 210,000 two-dimensional, and 310,000 three-dimensional systems). A validation set is generated following the same distribution at 10% of the training set size. This dataset size was chosen based on scaling experiments (Appendix E.6). Data characteristics, including spatial biases in vector field sampling, are analyzed in Appendix B.4.

B.4. Data Characteristics: Vector Field Magnitude Near Boundaries

Understanding the characteristics of the synthetic training data is crucial for interpreting model behavior. This section examines how vector field magnitudes vary spatially within the bounding boxes defined by trajectories.

Magnitude Distribution by Distance to Boundaries. During data generation, vector field samples are drawn uniformly within the bounding box defined by the trajectories. This introduces a systematic bias: vector field magnitudes tend to be higher near the boundaries of the bounding box, and variability is also most pronounced near boundaries. Figures 3, 4, and 5 show this pattern across dimensions 1, 2, and 3.

1D Outlier Behavior. The 1D case exhibits distinct behavior compared to 2D and 3D. According to the mean statistics, the magnitude is highest at the borders but also increases in the center of the interval. This pattern is caused by outlier ODEs where the vector field crosses the x-axis, resulting in instantaneous magnitude changes (Figure 6). In contrast, 2D and 3D cases show more consistent patterns without such outlier influence.

Implications for Training. As the dimension increases, a larger proportion of uniformly sampled points lie near the boundaries of the bounding box (a consequence of the curse of dimensionality). Meanwhile, the trajectories remain concentrated near the center by construction (since the initial conditions are sampled from $\mathcal{N}(0, I)$). This creates a growing mismatch between where the vector field is sampled for training and where trajectory observations are concentrated. To mitigate this, we employ a mixed sampling strategy: 50% of training locations are drawn from observed trajectory points, and 50% are sampled uniformly within the bounding box.

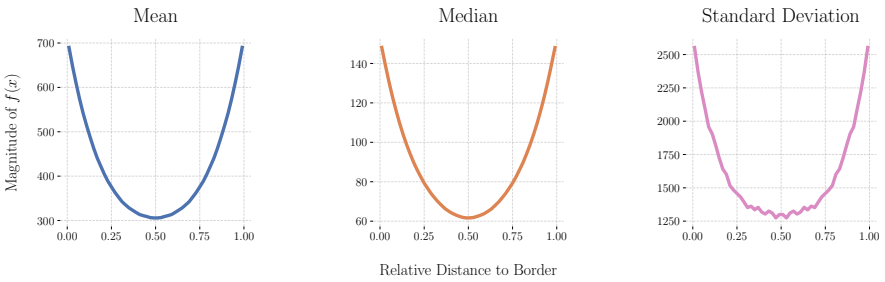


Figure 5. Magnitude statistics of vector field points as a function of relative distance to bounding box borders for 3D ODEs.

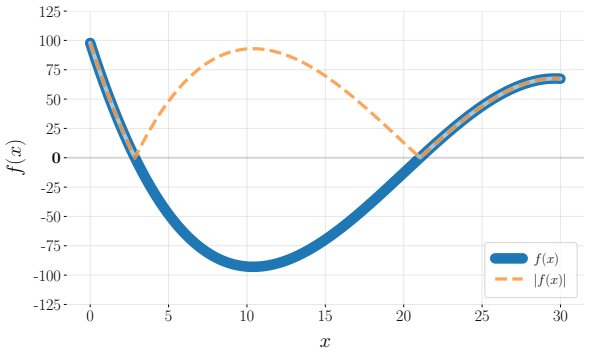


Figure 6. Example 1D outlier ODE. Vector field (blue) and magnitude of vector field (orange). The magnitude changes abruptly when the vector field crosses the x-axis, creating the distinctive pattern observed in 1D statistics.

C. Architecture and Training Details

This appendix provides comprehensive details on the `FIM-ODE` architecture, normalization schemes, and training procedures.

C.1. Architecture Specifications

The implemented model has an embedding dimension of $n = 256$. The encoder stack consists of 2 transformer layers, while the decoder stack uses 8 layers. Both the encoder and decoder blocks use multi-head attention with 8 heads and apply the Gaussian Error Linear Units (GELU) activation function. The final MLP consists of three layers with a hidden dimension of 1024.

The model supports systems up to dimension $d \leq 3$ through a padding scheme: when the model receives input with a system of lower dimensionality, the input is padded with zeros in the remaining dimensions.

C.2. Normalization Schemes

To generalize across systems on different spatial and temporal scales, the model uses two instance normalization schemes.

Spatial Normalization: Reversible Instance Normalization. All observations are normalized per dimension using:

$$\mathrm{IN}(\mathbf{y}(t)) = \frac{\mathbf{y}(t) - \mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}} \tag{10}$$

where $\mu_{\mathbf{Y}}$ and $\sigma_{\mathbf{Y}}$ are the mean and standard deviation computed per dimension over the truncated observation set $\mathbf{Y}$ (excluding the last observation of each trajectory due to feature extraction).

The model is trained to predict the normalized flow field:

$$\frac{d\,\mathrm{IN}(\mathbf{y}(t))}{dt} = \hat{\mathbf{f}}_\theta(\mathrm{IN}(\mathbf{y}(t))) \tag{11}$$

To return predictions to the original space, the chain rule yields:

$$\frac{d\mathbf{y}(t)}{dt} = \sigma_{\mathbf{Y}} \cdot \hat{\mathbf{f}}_\theta(\mathrm{IN}(\mathbf{y}(t))) \tag{12}$$

Thus, denormalization is implemented by scaling the predicted vector field by $\sigma_{\mathbf{Y}}$.

Temporal Normalization: Delta Log Centering. The times are normalized based on Δt. The Δt values are centralized around a target value $\Delta\tau_{target} = 0.01$ while ensuring all $\Delta t > 0$:

$$C(t) = \gamma \cdot t, \quad \gamma = \Delta\tau_{target} \cdot \exp(-\mu_{\log(\Delta t_{\mathbf{Y}})}) \tag{13}$$

where $\mu_{\log(\Delta t_{\mathbf{Y}})}$ is the mean of the logarithms of the time gaps in observation time.

Temporal denormalization also reduces to multiplication by γ:

$$\frac{d\mathbf{y}(C(t))}{dt} = \hat{\mathbf{f}}_\theta(\mathbf{y}(C(t))) \cdot \gamma \tag{14}$$

C.3. Training Procedures

Our training procedure extends the `FIM-SDE` methodology (Seifner et al., 2025a) with ODE-specific components.

Vector Field Training Mode. The primary training mode evaluates the model at sampled locations and compares predictions against ground truth flow values. During each training iteration, we sample locations $\mathbf{L}$ using a 50-50 split:

$$\mathbf{L} = \mathbf{L}_{traj} \cup \mathbf{L}_{random}, \quad \text{where} \quad |\mathbf{L}_{traj}| = |\mathbf{L}_{random}| \tag{15}$$

where $\mathbf{L}_{traj}$ contains all observed trajectory points (from the K trajectories used as context) and $\mathbf{L}_{random}$ contains an equal number of uniformly sampled points within the bounding box of the trajectories (expanded by 20%). This dual sampling strategy ensures the model learns both trajectory reconstruction and generalization to nearby regions of state space.

Trajectory / neural ODE Training Mode. This training method is based just on trajectories and requires no knowledge of the underlying vector field. We use it for finetuning the pre-trained model to specific ODEs. It is outlined in Appendix D.

Normalized Training. All loss computations occur in normalized space (after applying reversible instance normalization and delta log centering). This is critical for handling systems with vastly different scales and ensures the uncertainty weighting operates on comparable magnitudes across all systems.

Training Configuration. We train using the AdamW optimizer (Loshchilov & Hutter, 2019) with weight decay 1×10^{-4}, learning rate 1×10^{-5}, batch size 64, 10% dropout, and gradient clipping (max norm 10). The number of trajectories per batch is randomly varied between 1 and 9 to teach the model to handle variable context sizes. Training was performed over five days on four NVIDIA A40 GPUs (48 GB each), optimizing approximately 13 million parameters (8M for FIM-ODE, 5M for uncertainty estimation).

Pretraining Dataset. The pretraining dataset contains 600,000 polynomial ODE systems (80K 1D, 210K 2D, 310K 3D), with a 10% validation split following the same distribution. This is comparable in scale to the `FIM-SDE` pretraining (Seifner et al., 2025a) but adapted for the deterministic setting with multiple trajectories per system. Ablations on dataset size (Appendix E.6) show that performance scales with data but with diminishing returns beyond 100,000 systems for the tested model capacity.

C.4. Training Statistics

This section provides detailed training statistics for the `FIM-ODE` model, including loss curves, uncertainty estimates, and gradient norms during pretraining.

Loss Evolution. Figure 7 shows the weighted $\mathcal{L}_1$ training and validation losses throughout pretraining. The training losses are computed using partially randomized vector field locations and random subsets of trajectories, while validation losses are evaluated on all vector field locations conditioned on all available trajectories. This accounts for the smoother appearance of the validation curve. Although the loss curves indicate that further training could yield marginal improvements, the rate of improvement has slowed considerably by the end of training.

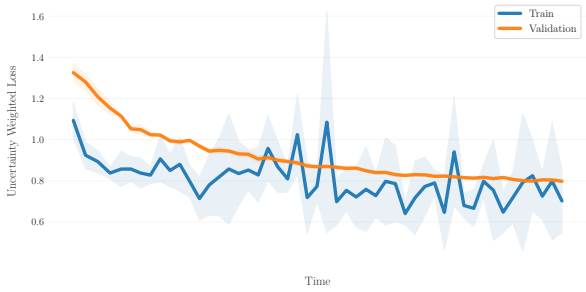


Figure 7. Weighted $\mathcal{L}_1$ training and validation losses during pretraining. Loss is weighted according to learned uncertainty estimates.

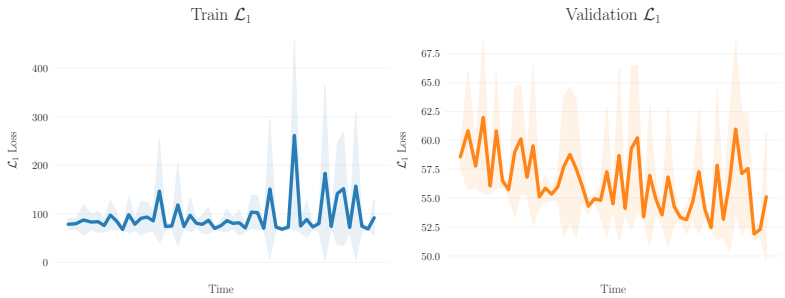


Figure 8. Unweighted $\mathcal{L}_1$ training (left) and validation (right) losses during pretraining.

Unweighted Loss Comparison. Figure 8 presents the unweighted $\mathcal{L}_1$ losses. The training $\mathcal{L}_1$ is consistently higher than the validation $\mathcal{L}_1$, which is explained by the different conditioning: during training, the model accesses only a subset of trajectories, whereas during validation all trajectories are provided. Since vector field predictions empirically improve when conditioned on more input trajectories, the validation loss is correspondingly lower.

Uncertainty and Gradient Norms. Figure 9 shows the evolution of uncertainty estimates and gradient norms. The uncertainty estimate, which the model uses to weigh loss terms, decreases during training, indicating increasing confidence in predictions. The gradient norm exhibits an upward trend during training, though gradient clipping (max norm 10) prevents this from destabilizing training. The increasing gradient magnitude in later training stages suggests the model is making finer adjustments to capture subtle features of the flow fields.

D. Finetuning `FIM-ODE`

After pretraining `FIM-ODE` we finetune the model to specific ODE by performing neural ODE (c.f., (Chen et al., 2018)) training based on the ODE's observed trajectories. As input we have one or more discrete trajectories $(x_i, t_i)_{i=1,\dots,\ell}$. The model uses all of these trajectories as the context to infer an ODE which is solved numerically for a collection of initial conditions, each defining a predicted trajectory $(\hat{x}_i)_{i=0,\dots,n_{\text{steps}}}$, on the context trajectories. After n_{steps} steps $\hat{x}_i$ (synchronised with the observed trajectory's time grid), the MAE

$$\frac{1}{n_{\text{steps}}} \sum_{i=1}^{n_{\text{steps}}} \|\hat{x}_i - x_i\|_1$$

is computed for each initial condition. We then add all the MAEs for the different initial conditions to obtain the final neural ODE training loss.

In all finetuning experiments the initial conditions are chosen as regularly interspaced on the time grid, starting with the trajectory's first observed point and we use $n_{\text{steps}} = 25$. We ensure that integration intervals of neighbouring initial conditions overlap by approximately half of their length by choosing $n_{\text{IC}} = \left\lfloor \frac{2\ell}{n_{\text{steps}}} \right\rfloor$ equally spaced initial conditions. The only exception to this is the FitzHugh–Nagumo task as in this task there are less than 25 observations in the context trajectory.

The MoCap tasks' training differs from the Van der Pol and FitzHugh–Nagumo tasks' finetuning in that there are multiple

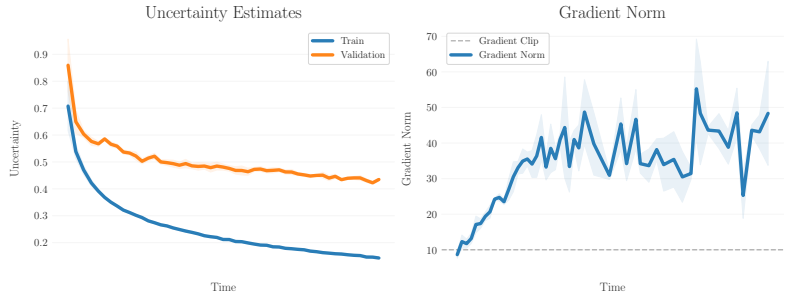


Figure 9. Uncertainty estimates for training and validation (left), and gradient norm evolution (right) during pretraining.

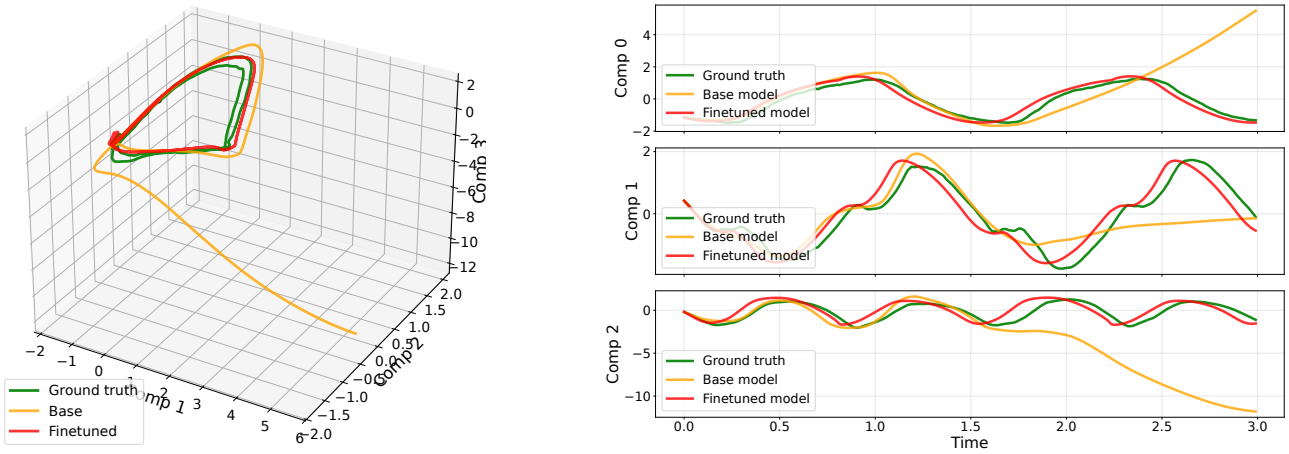


Figure 10. Finetuning task: MoCap 35 with short context length

context trajectories as well as separate validation and testing sequences. We choose the best model based on the neural ODE loss for the validation trajectories. For VDP and FHN, the training and validation trajectories are identical.

In less data-poor settings than VDP and FHN, such as the MoCap tasks, we find it useful to regularize finetuning by injecting noise at each trajectory step, or also at the initial conditions. Throughout all MoCap finetuning tasks, we employ the same noise level of $\sigma_i = \frac{\Delta t}{5}$ where $\Delta t = t_{i+1} - t_i$ is the time step in ODE solving.

For all tasks we trained for either 200 or 800 epochs although in most cases as few as 20 can have very good results. Finetuning on few trajectories requires little memory and does not benefit much from availability of a GPU, enabling practitioners to improve `FIM-ODE`'s inference capabilities at low cost. We outperform the neural ODE trained by (Hegde et al., 2022) in all tasks, proving that `FIM-ODE`'s pretraining is an effective way of amortizing neural ODE training.

E. Additional Experiments

E.1. Generalization Results: $R^2 > 0.8$ case

The R^2 score (coefficient of determination) measures trajectory reconstruction quality as $R^2 = 1 - \frac{\sum_i (x_i - \hat{x}_i)^2}{\sum_i (x_i - \bar{x})^2}$, where $\hat{x}_i$ are predictions, x_i are ground truth values, and $\bar{x}$ is the mean. For multi-dimensional systems, we use the variance-weighted R^2, where each dimension's R^2 is weighted by its proportion of total variance across all dimensions.

Tables 2 of the main text and Table 6 suggest that the relative performance depends strongly on how strict the success criterion is. At the stringent threshold $R^2 > 0.9$ (Tables 2), the two models are broadly comparable, with no consistent winner across noise and subsampling settings. However, when we relax the threshold to $R^2 > 0.8$ (Table 6), `FIM-ODE` shows a clearer edge. Without subsampling ($\rho = 0$), `FIM-ODE` increases the success rate from 35.2% to 41.8% at $\sigma = 0$, from 32.8% to 41.8% at $\sigma = 0.03$, and from 34.4% to 38.5% at $\sigma = 0.05$. Under subsampling ($\rho = 0.5$), the two methods are closer, but `FIM-ODE` remains competitive and often improves robustness to noise, for example 39.3% versus 33.6% at

($\sigma = 0.05$). Overall, these results indicate that FIM−ODE more frequently captures the coarse structure of the dynamics well enough to yield useful, though not perfect, long horizon generalisation, which becomes visible once the evaluation allows moderate errors.

Table 6. Trajectory Generalization Task on ODEBench: Case $\{R^2 > 0.8\}$

Method	$\rho = 0.0$			$\rho = 0.5$		
	$\sigma = 0.0$	$\sigma = 0.03$	$\sigma = 0.05$	$\sigma = 0.0$	$\sigma = 0.03$	$\sigma = 0.05$
ODEformer	35.2%	32.8%	34.4%	35.2%	41.0%	33.6%
FIM−ODE	41.8%	41.8%	38.5%	36.1%	40.2%	39.3%

E.2. Investigation of the Influence of the Noise Seed for the VdP and FHN oscillators

The Van der Pol (VDP) and FitzHugh-Nagumo (FHN) benchmark results in Section 5.2 depend strongly on the specific noise realization and choice of observation time points. To quantify this sensitivity, we compared the original benchmark configurations against 100 randomly sampled datasets with the same number of observations.

Experimental Setup. For each system, we generated 100 independent trials varying the noise seed (for VDP Task 1 and FHN) or both the time grid and noise seed (for VDP Task 2), while maintaining all other experimental parameters fixed:

- **VDP Task 1**: 50 uniformly-spaced time points with independent noise realizations ($\sigma^2 = 0.05$)
- **VDP Task 2**: 50 randomly-sampled non-uniform time points with independent noise realizations ($\sigma^2 = 0.05$)
- **FHN**: 19 training observations with independent time grids and noise realizations ($\sigma^2 = 0.025$)

For each configuration, we evaluated the zero-shot FIM−ODE model and computed test-set MSE over the forecast/missing data regions.

Results and Implications. Table 7 summarizes the comprehensive statistics across 100 independent trials for each task, and Figure 11 shows the best-performing random samples from this analysis. The results reveal striking task-dependent sensitivity.

Our analysis suggests that the pretrained FIM−ODE model exhibits substantial capability for these dynamical systems, but the single-instance evaluation protocol obscures the true performance distribution.

E.3. Synthetic Polynomial ODEs: Comparison with ODEformer

In this section, we provide detailed results comparing FIM−ODE against ODEformer (d'Ascoli et al., 2024) on synthetic polynomial ODE systems, evaluating both in-distribution performance (degree-3 polynomials matching the pretraining distribution) and out-of-distribution generalization (degree-6 polynomials).

In-Distribution Synthetic Polynomials. We first evaluate on 4,000 synthetic polynomial ODE systems with maximum degree 3, matching the pretraining distribution. Figure 12 shows that FIM−ODE substantially outperforms ODEformer across all dimensionalities in both reconstruction and generalization tasks. For reconstruction, FIM−ODE achieves 90% success rate (trajectories with $R^2 > 0.9$) compared to ODEformer's 65%. The generalization gap is even more pronounced: FIM−ODE reaches 26% while ODEformer achieves only 18.5%. This advantage stems from FIM−ODE's continuous neural operator representation, which naturally interpolates between observed states, whereas ODEformer's symbolic expressions may struggle with discrete token limitations.

Out-of-Distribution Generalization. To assess robustness beyond the training distribution, we evaluate both models on degree-6 polynomials which are well outside the degree-3 pretraining regime. Table 8 shows that while both models experience performance degradation, FIM−ODE maintains its advantage. Remarkably, in the single-trajectory reconstruction setting, FIM−ODE's performance drops by only 2.9 percentage points (90.2% → 87.3%), demonstrating surprising resilience to higher-order polynomial dynamics. ODEformer shows a larger degradation of 5.5 points. For generalization tasks, both models experience larger drops, consistent with the inherently harder task of extrapolating to unseen initial conditions with more complex dynamics.

Table 7. Performance statistics over 100 random noise/sampling realizations. The statistics reveal substantial variability across different random configurations, indicating strong dependence on the specific noise realization and choice of observation time points.

Task	**Mean MSE**	**Median MSE**	**Std Dev**	**Min / Max MSE**
VDP Task 1	0.575	0.340	0.687	0.017 / 4.756
VDP Task 2	2.511	2.226	1.699	0.129 / 8.462
FHN	1.871	1.171	1.988	0.041 / 11.502

Table 8. Out-of-distribution performance on degree-6 polynomials. Values show % of trajectories with $R^2 > 0.9$. Models were trained only on degree-3 polynomials.

Method	Degree 3	Degree 6	Δ
Reconstruction (1 trajectory)			
`FIM-ODE`	90.2	87.3	-2.9
`ODEformer`	65.0	59.5	-5.5
Generalization (1 trajectory)			
`FIM-ODE`	26.0	15.0	-11.0
`ODEformer`	18.5	11.3	-7.2

E.4. Discretization Ablations

We evaluate `FIM-ODE`'s ability to handle different trajectory discretizations and varying numbers of input trajectories, testing its neural operator capabilities on in-distribution polynomial ODEs.

Experimental Setup. We test the model on 4,000 newly generated polynomial ODE systems (maximum degree 3) with varying numbers of trajectory points $n_{points} \in \{50, 100, 200, 250\}$ and varying numbers of input trajectories $K \in \{1, 9, 12\}$. The choice of n_{points} spans the range seen during training (50–200 points) and extends beyond it (250 points) to test extrapolation. Similarly, K ranges from the minimum (1) to maximum (9) seen during training, plus a test case beyond the training distribution (12 trajectories).

Results: Discretization Effects. Figure 13 shows reconstruction performance (percentage of trajectories with $R^2 > 0.9$) across all combinations of discretization levels and trajectory counts. For single-trajectory reconstruction, finer discretization provides marginal benefits. However, for 9 and 12 trajectories, performance slightly decreases as the number of points increases. This counter-intuitive result suggests that with more context trajectories available, the model can effectively interpolate the flow field even from coarser discretizations, and overly fine discretizations may introduce unnecessary complexity.

Dimension-Specific Performance. Figure 14 presents dimension-specific reconstruction results across all tested configurations. Each cell shows the percentage of d-dimensional trajectories achieving $R^2 > threshold$ for varying thresholds. A consistent pattern emerges: higher-dimensional trajectories are harder to reconstruct across all tested scenarios. This reflects the increased complexity of capturing coupled dynamics in higher dimensions and the curse of dimensionality in sampling the state space.

E.5. Vector Field Quality Metrics

While trajectory-based evaluation is our primary metric, we also assess vector field prediction quality directly using RMSE and cosine similarity computed on the 10,000 vector field samples generated for each ODE system.

Aggregated Results. Figures 15 and 16 show vector field prediction quality across different discretizations and trajectory counts. Both metrics indicate that providing more input trajectories substantially improves vector field estimation. The number of trajectory points has a comparatively smaller impact than observed in trajectory reconstruction, likely because overall vector field error is dominated by regions with large magnitude, while fine details critical for accurate long-term trajectory integration have less weight in these aggregate metrics.

Per-Dimension Analysis. Figures 17 and 18 break down vector field quality by dimension for the standard 200-point discretization. An interesting observation is that while vector field predictions improve with more input trajectories, the

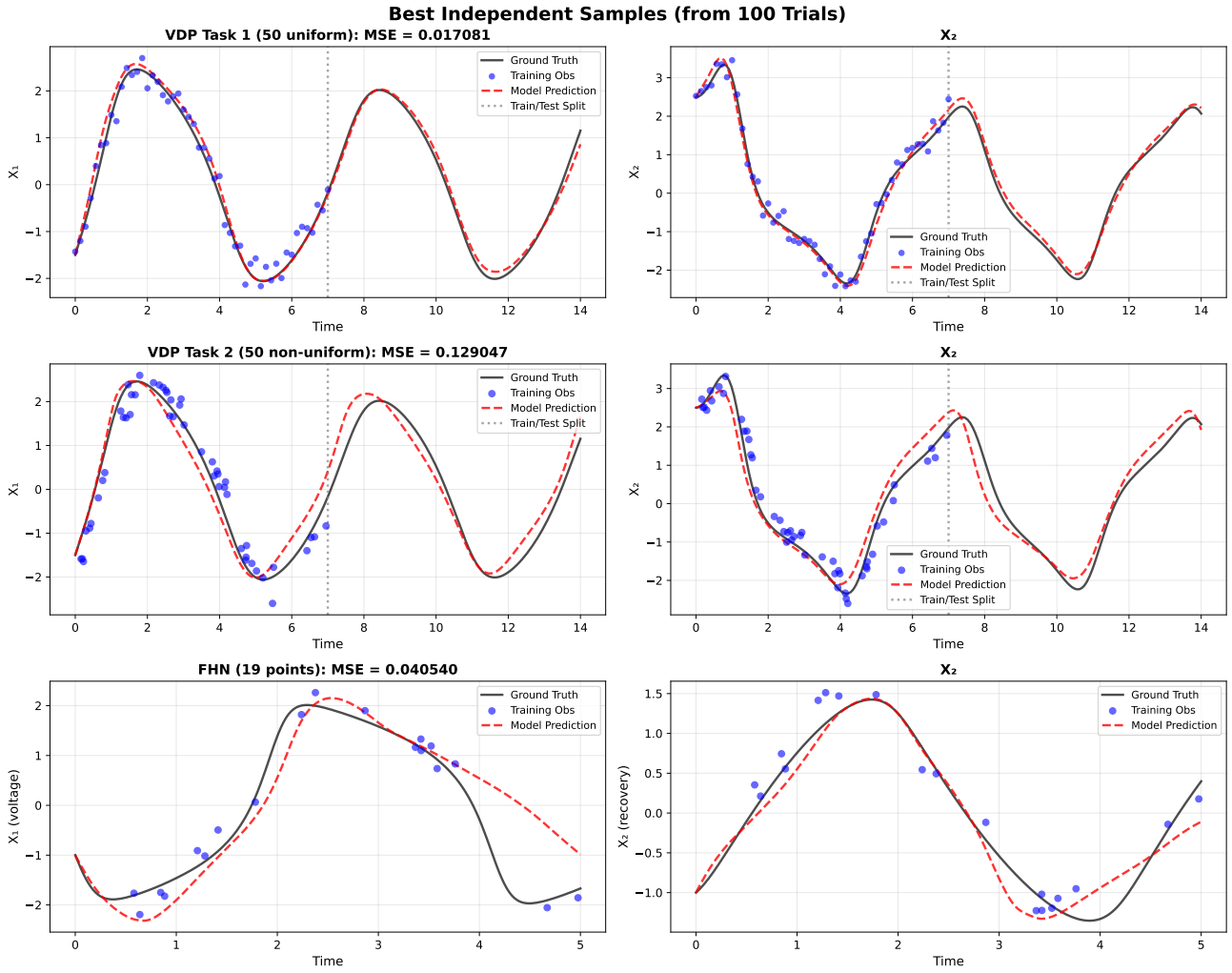


Figure 11. Best-performing random samples from 100 independent noise/sampling trials for each benchmark. Blue dots indicate training observations, black lines show ground truth, and red dashed lines show model predictions.

benefit diminishes with increasing dimension. For 1D systems, the median RMSE decreases by a factor of approximately 4 when given 9 trajectories instead of 1. This improvement factor reduces to 1.8 in 2D and 1.3 in 3D. This suggests that higher-dimensional systems require substantially more trajectory coverage to fully constrain the flow field, a consideration for future work targeting higher dimensions.

E.6. Impact of Dataset Size

We investigate how pretraining dataset size affects model performance by training four variants of `FIM-ODE` on datasets of 10,000, 50,000, 100,000, and 200,000 synthetic polynomial ODEs. To isolate the effect of dataset size, we use a smaller model architecture (approximately 1.3M parameters including uncertainty estimation) and train each variant for approximately 10 hours on a single NVIDIA A40 GPU.

Polynomial ODE Performance. Figure 19 shows reconstruction and generalization performance on in-distribution polynomial test data. Larger training datasets consistently produce better performing models, but improvements diminish beyond 100,000 training examples. This suggests diminishing returns for the tested model capacity, indicating that further scaling would require proportionally larger model architectures.

ODEBench Performance. Figures 20 and 21 show ODEBench performance across dataset sizes. For reconstruction, models trained on 200,000 systems perform comparably to those trained on 50,000 or 100,000 systems, with only minimal

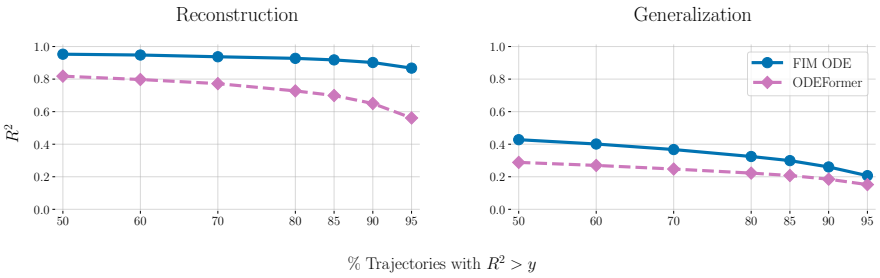


Figure 12. Comparison of `FIM-ODE` and `ODEformer` on polynomial ODEs (degree ≤ 3). Left: reconstruction performance. Right: generalization to new initial conditions. Both evaluated across dimensions $d \in \{1, 2, 3\}$.

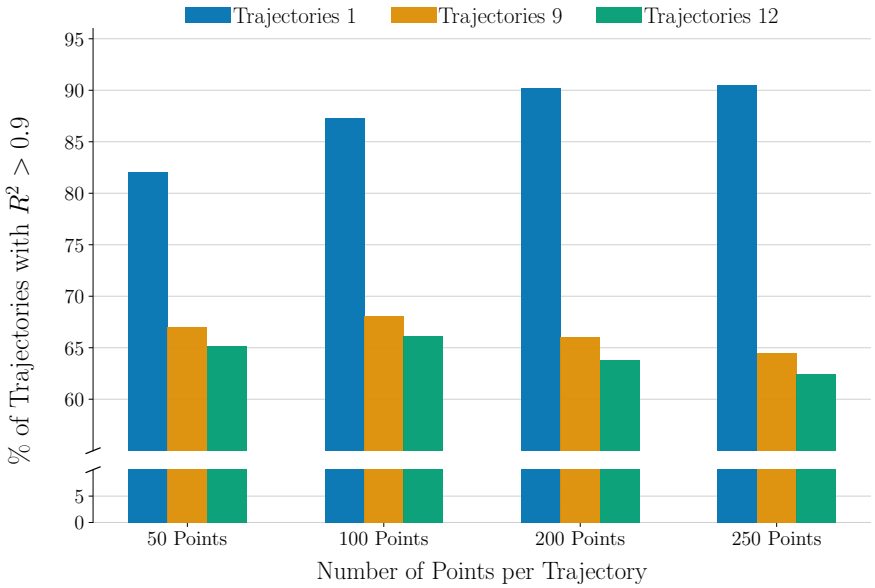


Figure 13. Reconstruction performance for different trajectory discretizations (n_{points}) and numbers of input trajectories (K). Higher is better.

differences on uncorrupted inputs. This suggests that even moderate-scale pretraining (50,000–100,000 systems) provides sufficient coverage of polynomial ODE dynamics for zero-shot transfer to real-world systems. For generalization, larger datasets provide more consistent benefits, particularly under data corruption.

Implications. These results demonstrate that `FIM-ODE`'s foundation model approach benefits from scale, but also that moderate-scale pretraining (on the order of 100,000 diverse systems) is sufficient for strong zero-shot performance. The diminishing returns suggest that future improvements should focus on jointly scaling both data and model capacity, following established neural scaling laws.

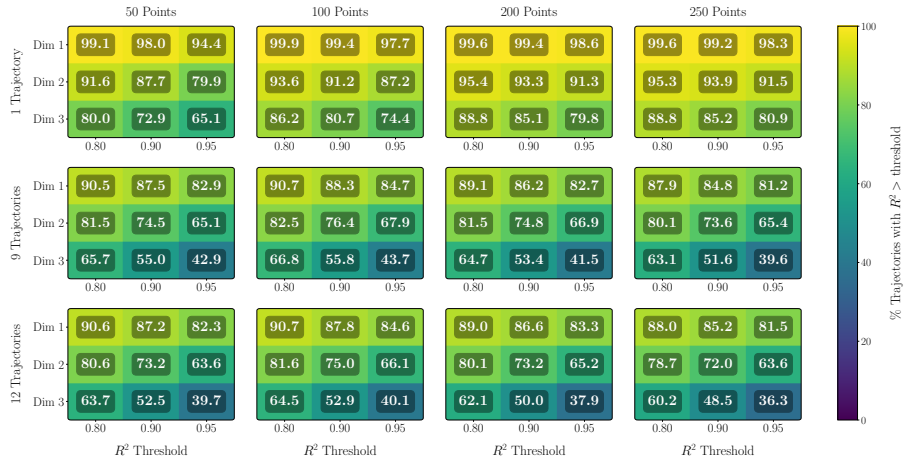


Figure 14. Reconstruction performance as percentage of trajectories with $R^2 > threshold$, shown separately for each dimension. Brighter colors indicate better performance. Results shown for all tested model inputs (combinations of n_{points} and K).

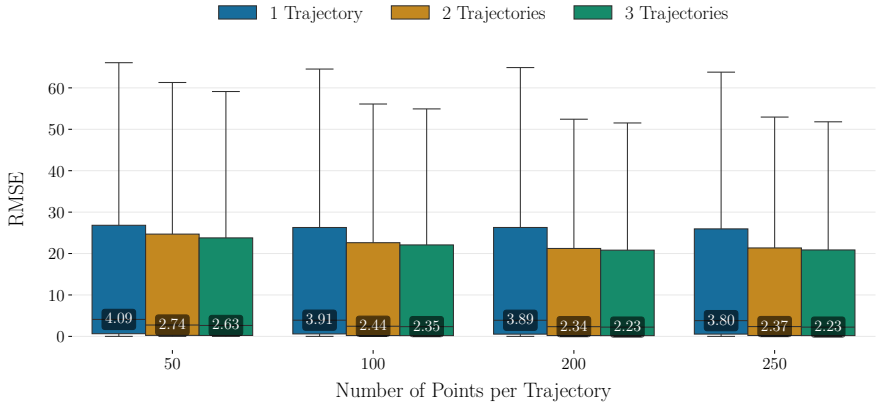


Figure 15. RMSE of vector field predictions for different trajectory discretizations and numbers of input trajectories. Lower is better.

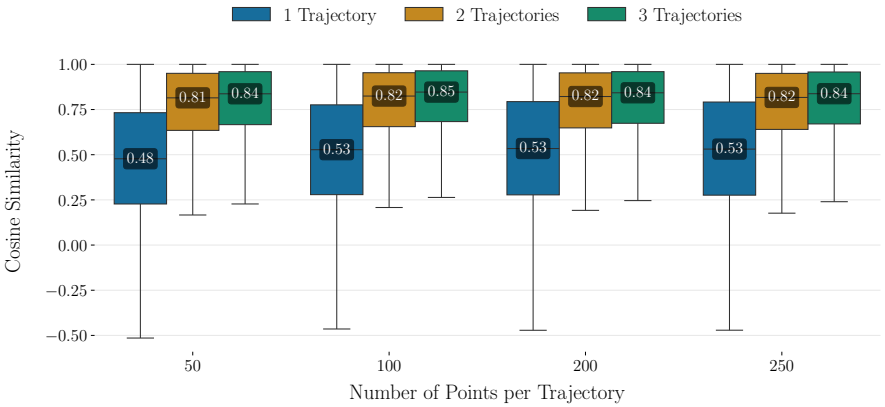


Figure 16. Cosine similarity between predicted and true vector fields for different trajectory discretizations and numbers of input trajectories. Higher is better.

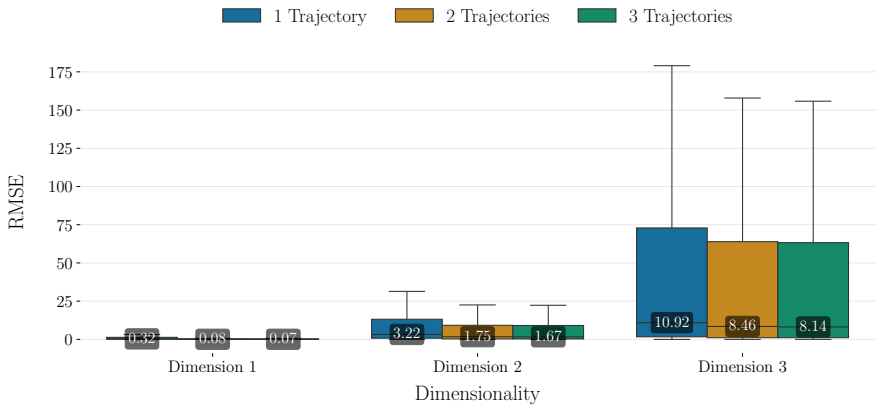


Figure 17. RMSE of vector field predictions per dimension for 200-point trajectory discretization. Lower is better.

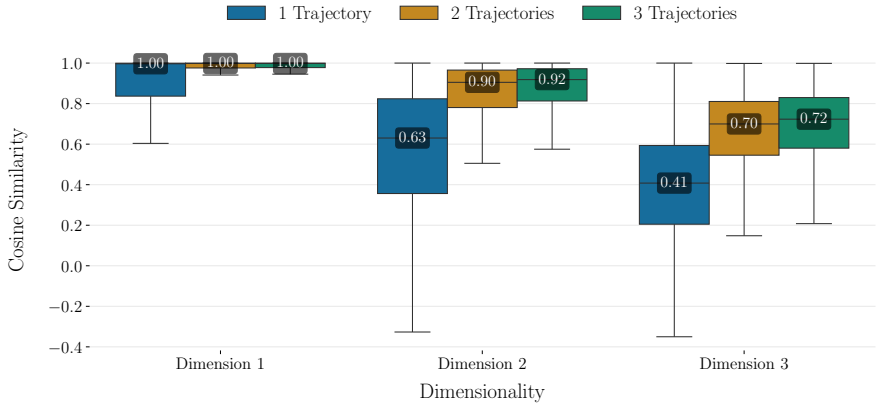


Figure 18. Cosine similarity between predicted and true vector fields per dimension for 200-point trajectory discretization. Higher is better.

Figure 19. Reconstruction (left) and generalization (right) performance as a function of training dataset size. Performance measured as percentage of trajectories with $R^2 > 0.9$ on in-distribution polynomial ODEs.

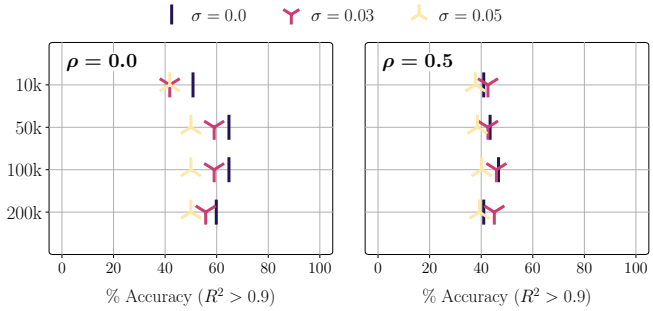


Figure 20. ODEBench reconstruction performance for models trained on 10k, 50k, 100k, and 200k polynomial ODEs. σ denotes Gaussian noise and ρ denotes the dropout ratio used for irregular sampling.

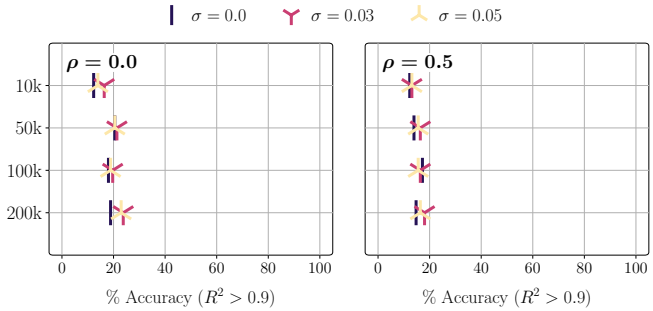


Figure 21. ODEBench generalization performance for models trained on 10k, 50k, 100k, and 200k polynomial ODEs. σ denotes Gaussian noise and ρ denotes the dropout ratio used for irregular sampling.