MANIFOLD-CONSTRAINED GAUSSIAN PROCESS INFERENCE FOR ONE-SHOT LEARNING OF UNKNOWN ORDINARY DIFFERENTIAL EQUATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning unknown ordinary differential equations (ODEs) from a single trajectory of scarce, noisy data is challenging, especially with partial observability. We introduce MAGI-X, an integration-free framework that couples a neural vector field with a Gaussian process prior over trajectories and enforces ODE consistency via a GP manifold constraint, thereby circumventing traditional numerical integration. Across canonical examples (FitzHugh–Nagumo, Lotka–Volterra, and Hes1), MAGI-X achieves better accuracy in both fitting and forecasting while requiring comparable or less computation time than benchmark methods NPODE and Neural ODE, with runtime scaling linearly in state dimension. MAGI-X offers a practical solution for *partially observed* systems without bespoke priors or imputation heuristics, where existing methods struggle. The GP posterior further yields calibrated uncertainty, and experiments demonstrate robustness across initial conditions. We show practicality on seasonal flu data with rolling multi-week forecasts from noisy signals. These properties establish MAGI-X as a fast, accurate, and robust tool for data-driven discovery of nonlinear dynamics from a single noisy trajectory.

1 INTRODUCTION

Systems of coupled ordinary differential equations (ODEs) are a fundamental tool for modeling complex mechanisms in science and engineering (FitzHugh, 1961; Nagumo et al., 1962; Lotka, 1932; Hirata et al., 2002). A well-known example is infectious disease prediction, where compartmental models such as SIR have long provided robust and interpretable descriptions of epidemic spread (Kermack and McKendrick, 1991). However, during pandemics or in multi-year endemic cycles with evolving interactions, simple compartmental models often fall short of capturing the true dynamics (Arik et al., 2020; Karlen, 2020; Yang et al., 2021b; Zou et al., 2020). Building more accurate ODE models by hand requires substantial domain expertise and repeated trial-and-error tuning. Similar challenges arise in systems biology and many other domains where the governing equations are only partially known (Hirata et al., 2002). These difficulties motivate a data-driven approach to learn ODE dynamics directly from one-single observation trajectories, especially when the true mechanisms are unknown or too complex to postulate a priori.

Problem formulation. Consider the dynamics governed by a system of ODEs:

$$\dot{x}(t) = f^*(x(t), \theta, t), \quad t \in [0, T], \tag{1}$$

where $x(t) = (x_1(t), \ldots, x_D(t))^\top \in \mathbb{R}^D$ is the state of the D-component system at time t, $\dot{x}(t) \in \mathbb{R}^D$ its time derivative, and f^* the unknown vector field with parameters θ. The state at time t is obtained by integration; in one dimension,

$$x(t) = x_0 + \int_0^t f^*(x(s), \theta, s)\, ds, \tag{2}$$

with initial condition $x_0 = x(0)$. We assume f^* is *completely unknown*, with no prior information about its form or parameters. Observations are noisy samples $y(\tau) = x(\tau) + \epsilon(\tau)$ at discrete times τ from one single trajectory, where ϵ denotes measurement noise. Given only these observations, the goal is to infer the *black-box* function f^*.

Related work. Traditional approaches assume a parametric form for f^* and estimate θ by minimizing misfit between simulated and observed trajectories, requiring repeated numerical integration (Bard, 1975; van Domselaar and Hemker, 1975; Benson, 1979). To avoid repeated solves, gradient-matching methods align derivative estimates from smoothed data with those given by f^*, using splines (Varah, 1982; Ramsay et al., 2007), RKHS models (González et al., 2014; Niu et al., 2016), or Gaussian processes (GPs) (Calderhead et al., 2009; Dondelinger et al., 2013; Wang and Barber, 2014; Macdonald et al., 2015; Wenk et al., 2019). Unified GP formulations resolve inconsistencies and extend to higher dimensions (Yang et al., 2021a; Wenk et al., 2020), but typically requires a pre-specified functional form or basis.

More flexible methods remove parametric assumptions. NPODE learns f^* nonparametrically with a GP (Heinonen et al., 2018), while Neural ODE parameterizes f^* with a neural network (NN) and trains via adjoint sensitivity (Chen et al., 2018). These established strong baselines but suffer from the cost of repeated integration, particularly in stiff or partially observed systems. Integration-free alternatives include two-step gradient matching with nonparametric f^* (Heinonen and d'Alché Buc, 2014; Ridderbusch et al., 2020), distributional or Bayesian gradient matching for Neural ODEs (Treven et al., 2021; Bonnaffé and Coulson, 2022), and GP-based identification with guarantees from sparse time series (Batko et al., 2024). These avoid ODE solves but typically lack a unified probabilistic objective, limiting robustness under irregular sampling or partial observability. We focus our experiments on NPODE and Neural ODE as representative, state-of-the-art baselines for single-trajectory ODE learning, since they cover the two dominant paradigms (GP-based and neural network–based integration methods). Other approaches are either variations of these or operate under different assumptions, as discussed below.

Another line of work pursues explicit equation discovery (Brunton et al., 2016) or probabilistic solvers for known ODEs (Hennig et al., 2015; Schober et al., 2019). While these provide interpretability or uncertainty in integration, they either require a predefined basis for f^* or assume the ODE is known, making them less applicable to our setting of learning a completely unknown vector field from scratch.

There are also operator-learning approaches (e.g., DeepONet ((Lu et al., 2021)), Fourier Neural Operator ((Li et al., 2021)), and variants incorporating physics (Wang et al., 2021) that learn mappings from initial conditions to solution trajectories across many systems. However, these require extensive training data across system families and thus do not address our one-trajectory learning setting.

Positioning. Our proposed MAGI-X method unifies different perspectives. Like gradient matching, it *circumvents numerical integration*; like GP frameworks, it provides a probabilistic treatment of trajectories and partial observability; and by learning f^* with a neural network within a manifold-constrained GP objective, it unifies trajectory smoothing and vector-field learning in a single optimization, yielding robustness under sparse/irregular sampling and favorable scaling.

Our contribution. We propose MAGI-X, a cost-efficient and theoretically principled framework for learning unknown dynamical systems. Our contributions are:

- **Efficiency and scalability:** By avoiding integration, MAGI-X outperforms integration-based methods while requiring comparable or less amount of runtime (Table 1) while scaling linearly in system dimension (Table 5).
- **Robust coupled inference and GP augmentation:** By using a feedback loop that incorporates information from f^* in adapting the smoothing trajectory, MAGI-X improves the robustness over the two-step gradient constrained approach. By proposing a novel optimization algorithm that utilizes the generative mechanism inherently built in the Gaussian process for data augmentation, MAGI-X could avoid over-fitting of neural network even when the training data is sparse.
- **Principled uncertainty and partial-observations handling:** The GP prior yields uncertainty-aware trajectories and naturally imputes missing data, crucial in applications such as multi-year infectious disease forecasting.
- **Single-trajectory learning:** From *one* noisy trajectory of a single system, MAGI-X achieves strong fit and forecast accuracy and shows empirical evidence of learning the system's internal dynamics (faithful vector fields; see Sec. 3.4).This one-shot capability is a core novelty that enables stable forecasts in low-data settings.

2 METHOD OF MAGI-X

Intuition. We consider the setting where the derivative function f^* is unknown. Prior work (Heinonen et al., 2018; Ridderbusch et al., 2020) models f^* with a vector-valued GP defined over a grid of inducing locations, but this entails $\mathcal{O}((DM)^3)$ cost from repeated covariance inversions (with the rule-of-thumb $M \approx 10D$ (Loeppky et al., 2009)), which becomes prohibitive as D grows. We therefore parameterize f^* with a neural network and learn it by first-order optimization—avoiding these cubic costs while remaining compatible with our manifold constraint. This choice is not only practical (fast first-order optimization, automatic batching) but also theoretically appropriate for our use of a manifold constraint: we show a quantitative, uniform approximation guarantee for ReLU networks on the compact input set actually used in training, and we use it to argue that the manifold constraint we impose is feasible on any discretization. For completeness, GP preliminaries are included in App. A (Rasmussen and Williams, 2006). By coupling the NN f_θ with a GP prior on trajectories, our manifold-constrained objective jointly smooths $x(\cdot)$ and enforces ODE consistency without numerical integration.

Prior. Following the Bayesian framework, we can view the D-component system response $x(t)$ as a realization of the stochastic processes $X(t) = (X_1(t), \ldots, X_D(t))$. We model the *unknown* f^* by neural network, denoted as $\hat{f}^{NN}$, where its bias and weight parameters are denoted as θ, a realization of random variable Θ. The goal is to compute the posterior distribution of $X(t)$ and Θ, which requires first defining the prior and the data likelihood.

We assume a a non-informative prior on Θ, that is $\pi_\Theta(\theta) \propto 1$. For the stochastic process $X(t)$, we impose an independent GP prior on each component, that is,

$$X_d(t) \sim \mathcal{GP}(\mu_d, \mathcal{K}_d),\ t \in [0, T], \tag{3}$$

with mean function $\mu_d : \mathbb{R} \to \mathbb{R}$ and positive definite covariance function $\mathcal{K}_d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ parameterized by hyperparameters ϕ_d.

Likelihood. Let us denote the observations $y(\tau) = (y_1(\tau_1), \ldots, y_D(\tau_D))$ where $\tau_d = (\tau_{d,1}, \ldots, \tau_{d,N_d})$ is the set of N_d observation time points for the d-th component. We allow different components to be observed at different sets of time points, but none of the component can be completely missing, that is $N_d > 0\ \forall d$. For simplicity, we assume the observation noise for the d-th component are i.i.d. zero mean Gaussian random variable with variance σ_d^2. Also, the noise variance σ^2 is generally not known apriori, so a non-informative prior can also be assigned, that is $p(\sigma^2) \propto 1$. Then, the observation likelihood is

$$Y_d(\tau_d) | X_d(\tau_d) = x_d(\tau_d) \sim \mathcal{N}(x_d(\tau_d), \sigma_d^2 I_{N_d}). \tag{4}$$

Next, we introduce W that quantifies the difference between the time derivative $\dot{X}(t)$ of GP and the ODE imposed gradient $\hat{f}^{NN}(X(t), \theta, t)$ with any given θ,

$$W = \sup_{t \in [0,T], d \in \{1,\ldots,D\}} \left| \dot{X}_d(t) - \{\hat{f}^{NN}(X(t), \theta, t)\}_d \right|. \tag{5}$$

Having $W \equiv 0$, the ideal case where the derivative function f defined by θ is fully satisfied by $X(t)$, is to impose a *manifold constraint* on the GPs that model $X(t)$. However, W defined in equation 5 cannot be computed analytically since it is a supremum over an uncountable set. Thus, we approximate $W_\mathcal{T}$ by finite discretization on $\mathcal{T} = (t_1, \ldots, t_n) \subset [0, T]$,

$$W_\mathcal{T} = \max_{t \in \mathcal{T}, d \in \{1,\ldots,D\}} \left| \dot{X}_d(t) - \{\hat{f}^{NN}(X(t), \theta, t)\}_d \right|. \tag{6}$$

It follows that a computable closed-form likelihood associated with the manifold constraint is

$$\begin{aligned} &p\{W_\mathcal{T} = 0 \mid X(\mathcal{T}) = x(\mathcal{T}), \Theta = \theta\} \\ &= p\{\dot{X}(\mathcal{T}) = f(x(\mathcal{T}), \theta, t) \mid X(\mathcal{T}) = x(\mathcal{T}), \Theta = \theta\}. \end{aligned} \tag{7}$$

Since the time derivative of GP is also again GP with specific mean and covariance function (see supplementary materials for details), equation 7 is the p.d.f. of some multivariate Gaussian distribution.

Algorithm 1: MAGI-X

Input: (i) observations $y(\tau) = \{y_d(\tau_d)\}_{d=1}^D$; (ii) discretized time points $\mathcal{T} = (t_1, \ldots, t_n)$; (iii) neural network architecture with parameters θ.

Initialization:;

time standardization (see supplementary material for details);

for $d = 1, \ldots, D$ **do**

  Fit GP regression on $y_d(\tau_d)$ to identify hyperparameters ϕ_d and noise variance σ_d^2;

  Initialize $x_d(\mathcal{T})$ using the predictive mean of the trained GP evaluated at $\mathcal{T}$;

end

Initialize θ by optimizing equation 8 w.r.t. θ only, fixing $x(\mathcal{T})$ and $\{\sigma_d^2\}_{d=1}^D$ at initial values;

Optimization (block-wise updates alternating between θ, $x(\mathcal{T})$, and $\{\sigma_d^2\}_{d=1}^D$):;

for $l = 1, \ldots, L$ **do**

  Gradient ascent on θ with learning rate $\eta_l^{(\theta)} = 0.005\, l^{-0.6}$;

  Gradient ascent on $x(\mathcal{T})$ with learning rate $\eta_l^{(x)} = 0.05\,(500 + l)^{-0.6}$;

  Closed-form update for $\{\sigma_d^2\}_{d=1}^D$;

end

Return: optimized θ, $x(\mathcal{T})$, and $\{\sigma_d^2\}_{d=1}^D$;

Furthermore, $W_{\mathcal{T}} \to W$ monotonically as $\mathcal{T}$ becomes dense (Yang et al., 2021a), suggesting the choice of a finer discretization of $\mathcal{T}$ than the observation time points, where $|\mathcal{T}| > N_d,\ \forall d$. However, this leads to stronger emphasis on the GP prior equation 3 and the manifold constraint likelihood equation 7 in the posterior function of $X(t)$ and Θ, especially if we allow the cardinality of $\mathcal{T}$ to be arbitrarily large for a precise approximation. To balance out the influence from the observations and the discretization points, we introduce tempering parameters on the observation likelihood equation 4.

Posterior and Objective Function. We therefore have the following Maximum a Posteriori (MAP) estimation objective function (log-posterior function):

$$\begin{aligned}
&\arg\max_{\theta, x(\mathcal{T}), \sigma^2} \log p\{\Theta = \theta, X(\mathcal{T}) = x(\mathcal{T}) \mid W_{\mathcal{T}} = 0, Y(\tau) = y(\tau)\} \\
&\quad \propto \arg\max_{\theta, x(\mathcal{T}), \sigma^2} \underbrace{\log p\{X(\mathcal{T}) = x(\mathcal{T})\}}_{\text{equation 3}} + \underbrace{\log p\{W_{\mathcal{T}} = 0 \mid X(\mathcal{T}) = x(\mathcal{T}), \Theta = \theta\}}_{\text{equation 7}} + \\
&\qquad \underbrace{\sum_{d=1}^{D} \tfrac{|\mathcal{T}|}{N_d} \log p\{Y_d(\tau) = y_d(\tau) \mid X_d(\tau) = x_d(\tau), \sigma_d^2\}}_{\text{equation 4}},
\end{aligned} \tag{8}$$

where equation 3, equation 4, and equation 7 are all p.d.f. of some multivariate normal distributions.

Optimization procedure. MAGI-X uses two functional approximations: the manifold-constrained GP approximation of the trajectory and the neural network approximation of the derivative function f^*. This makes optimizing equation 8 difficult, so a robust initialization is essential. We first standardize time to ensure numerical stability across different ranges. Then, for each component $y_d(\tau_d)$, we fit an independent GP regression and initialize $x_d(\mathcal{T})$ with its predictive mean at $\mathcal{T}$. The GP hyperparameters $\{\phi_d\}_{d=1}^D$ and noise variances $\{\sigma_d^2\}_{d=1}^D$ are set via empirical Bayes. To initialize θ, we optimize equation 8 only over θ, holding $x(\mathcal{T})$ and $\{\sigma_d^2\}_{d=1}^D$ fixed. This matches the two-step approach of Ridderbusch et al. (2020), but with a neural model for f^*. However, the resulting dynamics can generalize poorly, weakening forecasts, requiring a subsequent *feedback loop* alternating between $x(\mathcal{T})$ and θ. For efficiency, $\{\phi_d\}_{d=1}^D$ remain fixed after initialization.

In the main optimization, we alternate updates of θ, $x(\mathcal{T})$, and σ^2. Before each θ update, block ascent on $x(\mathcal{T})$ is performed, yielding fresh input/output pairs $\{x(\mathcal{T}), \dot{x}(\mathcal{T})\}$ for gradient computation. The GP framework thereby acts as an implicit data-augmentation mechanism, producing infinitely many plausible, slightly noisy training pairs. Because these resemble samples from the trajectory posterior, we use a polynomially decayed learning rate $\eta_l = a(b + l)^{-\gamma}$ (e.g., $l^{-0.6}$ as in Algorithm 1) to encourage convergence. Moreover, each new set of pairs also serves as a moving validation signal, improving generalization and mitigating neural overfitting under sparse observations. Algorithm 1 outlines the procedure; numerical-stability details are in the supplement.

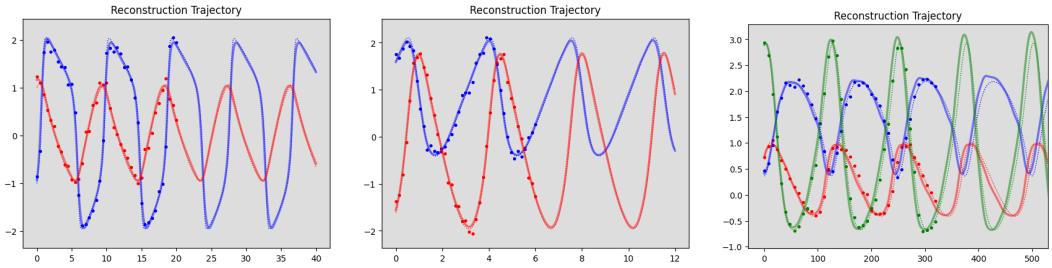
3 RESULTS

We evaluate MAGI-X on three systems: FitzHugh–Nagumo (FN) FitzHugh (1961); Nagumo et al. (1962), Lotka–Volterra (LV) Lotka (1932), and Hes1 Hirata et al. (2002). (Funtional forms are given in the Supplementary Materials) We analyze LV and Hes1 in log space since they are strictly positive. We first consider the fully observed case (Sec. 3.1), followed by the partially observed setting straightforwardly applies without imputation/bespoke priors (Sec. 3.2). We then demonstrate robustness to different ODE initial conditions (Sec. 3.3), examine the learned derivative vector field (Sec. 3.4), and the uncertainty quantification with calibration (Sec. 3.5). Finally, a real-world influenza data study (Sec. 3.6) will conclude this section. Our objective is to learn the *unknown* dynamics, so only noisy observations are fed to MAGI-X for inference; the parametric forms are not available to the method.

Groundtruth data are simulated by numerical integration. To generate noisy observations $y(\tau)$, we use 41 points with i.i.d. Gaussian noise $\sigma_d^2 = 0.1^2$ during the first half of the period, which we call the *fitting* phase. The second half is for the *forecasting* phase. See Figure 1 for illustration. We report trajectory root mean squared error (RMSE) across components and phases (fitting and forecasting). Numerical integration is only used for evaluation and forecast, while MAGI-X requires *no numerical integration* during fitting.

For comparisons, we include NPODE (Heinonen et al., 2018) and Neural ODE (Chen et al., 2018), both integration-based methods. NPODE uses its default GP and runs for 500 iterations. Neural ODE is matched to MAGI-X in NN architecture and employs the $\mathcal{O}(1)$-memory adjoint; due to its longer runtime, we cap training at 500 iterations for fairness. MAGI-X implementation details are given in App. C.2.

3.1 FULLY OBSERVED SYSTEM



(a) FN (b) LV (log-transformed) (c) Hes1 (log-transformed)

Figure 1: Comparison of the inferred trajectory (solid line) to the ground truth trajectory (dotted lines) after applying MAGI-X (Algorithm 1) on the 41-point fully observed data (circles).

Table 1: RMSE (mean ± std) over 100 runs with full observations. Runtime (RT, seconds) is shown next to each dataset block. Lower is better. Boldface red highlights the best performance.

		FN			LV			Hes1			
Phase	Model	x_1	x_2	**RT (s)**	x_1	x_2	**RT (s)**	x_1	x_2	x_3	**RT (s)**
Fitting	NPODE	0.24±0.02	0.06±0.02	36.02±1.82	0.10±0.02	0.13±0.03	36.01±1.42	1.78±1.72	1.44±1.45	2.40±1.88	85.23±0.66
	Neural ODE (512)	1.14±0.37	0.52±0.20	424.05±22.54	0.29±0.03	0.37±0.05	332.22±13.02	0.61±0.12	0.54±0.08	1.23±0.06	1770.79±490.00
	MAGI–X (512)	**0.11±0.02**	**0.05±0.01**	23.87±0.75	**0.05±0.01**	**0.05±0.02**	23.56±0.52	**0.09±0.06**	**0.06±0.04**	**0.12±0.10**	93.19±1.76
Forecasting	NPODE	0.26±0.07	0.08±0.04	36.02±1.82	0.16±0.10	0.21±0.13	36.01±1.42	1.93±1.87	1.57±1.63	2.62±2.04	85.23±0.66
	Neural ODE (512)	1.40±0.43	0.65±0.25	424.05±22.54	0.48±0.16	0.68±0.27	332.22±13.02	0.65±0.10	0.53±0.05	1.30±0.05	1770.79±490.00
	MAGI–X (512)	**0.12±0.04**	**0.05±0.02**	23.87±0.75	**0.13±0.12**	**0.17±0.17**	23.56±0.52	**0.37±1.31**	**0.14±0.13**	**0.34±0.33**	93.19±1.76

Across FN, LV, and Hes1, MAGI-X attains the lowest RMSE in both fitting and forecasting while remaining competitive in runtime. On FN/LV, MAGI-X is *both* more accurate and faster than NPODE (∼24 s vs. ∼36 s). On Hes1, NPODE's runtime is slightly lower (∼85 s vs. ∼93 s) but its errors are an order of magnitude larger, so MAGI-X still dominates the accuracy–time trade-off (Table 1). To rule out undertraining, we extended NPODE to 2000 iterations; the accuracy–time trade-off results were unchanged (App. D.1).

3.2 Partially Observed System

Table 2: Means and standard deviations of trajectory RMSEs over 100 runs with partial observations.

		FN		LV		Hes1		
Phase	Model	x_1	x_2	x_1	x_2	x_1	x_2	x_3
Fitting	MAGI-X (512)	0.21 ± 0.02	0.06 ± 0.02	0.06 ± 0.02	0.07 ± 0.03	0.12 ± 0.04	0.07 ± 0.02	0.15 ± 0.04
Forecasting	MAGI-X (512)	0.21 ± 0.04	0.06 ± 0.03	0.13 ± 0.10	0.19 ± 0.15	0.29 ± 0.14	0.17 ± 0.07	0.39 ± 0.18

MAGI-X also provides a full solution for systems with asynchronous observation times, a common scenario in practice caused by sensor outages or human error. Through the Bayesian GP framework, MAGI-X naturally handles such partial observations by principled probabilistic imputation, requiring no modification.

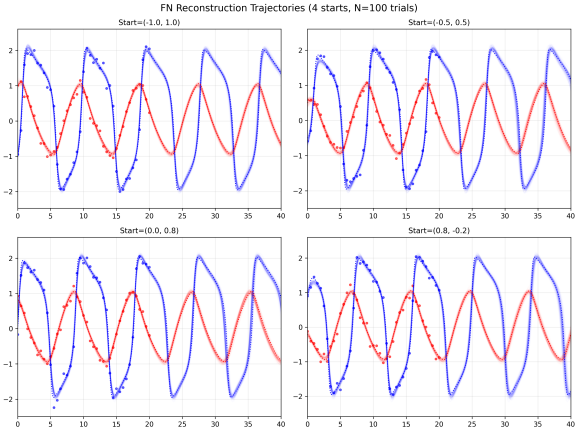
To test robustness against partially observed systems, we simulate noisy data in an extreme setting. For the two-component FN and LV systems, only one component is observed at each time point. To match the fully observed case, we keep 41 observations, with $\tau_1 = \{t_1, t_5, \ldots, t_{161}\}$ and $\tau_2 = \{t_3, t_7, \ldots, t_{159}\}$, and i.i.d. Gaussian noise with variance $\sigma_d^2 = 0.1^2$ is added to obtain noisy y. From 100 independent partially observed datasets, Table 2 reports mean and standard deviation of trajectory RMSEs. Results are comparable to the fully observed case (Table 1), showing MAGI-X maintains accuracy even with partial data.

We consider an even more challenging setting with fewer, irregular observations for Hes1. Specifically, $\tau_1 = \{t_5, t_9, t_{17}, t_{21}, \ldots, t_{161}\}$, $\tau_2 = \{t_1, t_9, t_{13}, t_{21}, \ldots, t_{157}\}$, and $\tau_3 = \{t_1, t_5, t_{13}, t_{17}, \ldots, t_{161}\}$, where only two of three components are observable at any time, observation times are uneven, and each component has only ~ 28 observations, one third fewer than before. Simulating 100 datasets with i.i.d. Gaussian noise $\sigma_d^2 = 0.1^2$, Table 2 shows RMSEs about $\sqrt{41/28} \approx 1.21$ worse than the fully observed case- precisely the expected degradation. This shows MAGI-X's robustness on partially observed systems, a setting no existing method can handle.

3.3 Starting Point Robustness

We evaluate MAGI-X's robustness to initial conditions on the FitzHugh–Nagumo system using four distinct starting points: $(-1.0, 1.0)$, $(-0.5, 0.5)$, $(0.0, 0.8)$, and $(0.8, -0.2)$. For each initialization, we generate 41 noisy observations per component and perform 100 independent trials.

Figure 2 shows consistent trajectory reconstruction across all starting points. The RMSEs remain stable: fitting errors are tightly clustered (0.11–0.12 for x_1, 0.04–0.05 for x_2), and forecasting performance shows similar consistency. This demonstrates that MAGI-X learns robust dynamical representations independent of initialization choice.



Start		Fitting	Forecasting
$(-1.0, 1.0)$	x_1	0.11 ± 0.02	0.12 ± 0.04
	x_2	0.04 ± 0.01	0.05 ± 0.02
$(-0.5, 0.5)$	x_1	0.12 ± 0.02	0.17 ± 0.07
	x_2	0.05 ± 0.01	0.07 ± 0.03
$(0.0, 0.8)$	x_1	0.11 ± 0.02	0.18 ± 0.06
	x_2	0.05 ± 0.01	0.07 ± 0.03
$(0.8, -0.2)$	x_1	0.12 ± 0.02	0.17 ± 0.07
	x_2	0.05 ± 0.01	0.06 ± 0.02

Figure 2: Robustness to starting points on the FitzHugh–Nagumo system. Left: trajectory reconstructions (dotted = ground truth; transparent solid = MAGI-X). Right: trajectory RMSEs (mean $\pm$ std over 100 runs).

3.4 Derivative Analysis

To assess learned dynamics beyond trajectory fit, we compare each method's we analyze the derivative fields by comparing predicted derivatives $\hat{f}(x)$ with the true field $f^*(x)$ across the state space.

Figure 3 shows error magnitude heatmaps of $|\hat{f}(x) - f^*(x)|_2$ on a 40×40 grid. For both FN and LV systems, MAGI-X achieves lower errors, especially near the training data. Figure 4 visualizes the learned vector fields, with arrow magnitude indicated by color. While all methods capture key structures such as the limit cycles, MAGI-X most closely matches the true system, particularly away from training observations, indicating better generalization. Table 3 reports the RMSE of the derivative components (dx_1/dt, dx_2/dt), averaged over 100 independent runs. MAGI-X consistently yields lower derivative errors.

Table 3: Derivative RMSEs for FitzHugh–Nagumo (FN) and Lotka–Volterra (LV) (mean ± std over 100 runs).

	FN			LV		
	MAGI-X	Neural ODE	NPODE	MAGI-X	Neural ODE	NPODE
dx_1/dt	6.46 ± 0.69	7.33 ± 1.03	7.82 ± 0.81	0.45 ± 0.05	0.87 ± 0.09	1.23 ± 0.11
dx_2/dt	0.20 ± 0.07	0.62 ± 0.75	0.39 ± 0.05	2.99 ± 0.12	3.16 ± 0.18	4.44 ± 0.22

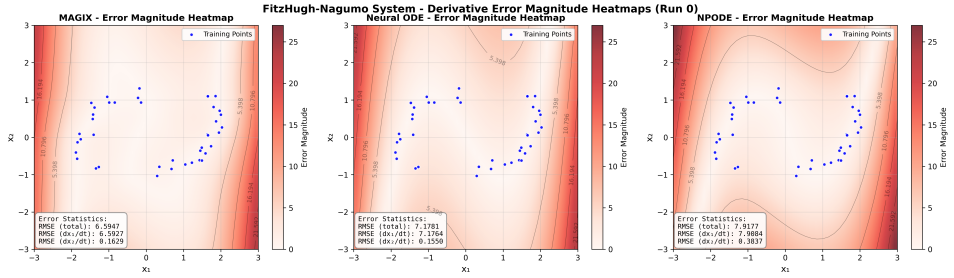


Figure 3: FN error heatmap: $\|\hat{f}(x) - f^*(x)\|_2$ across the state space (lighter is lower).

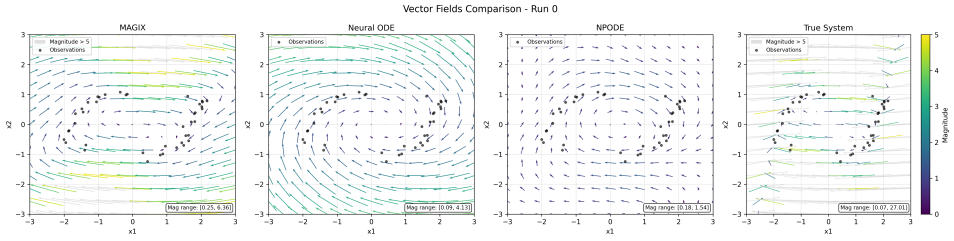


Figure 4: FN learned vector field vs. true dynamics. Arrows indicate flow direction and magnitude.

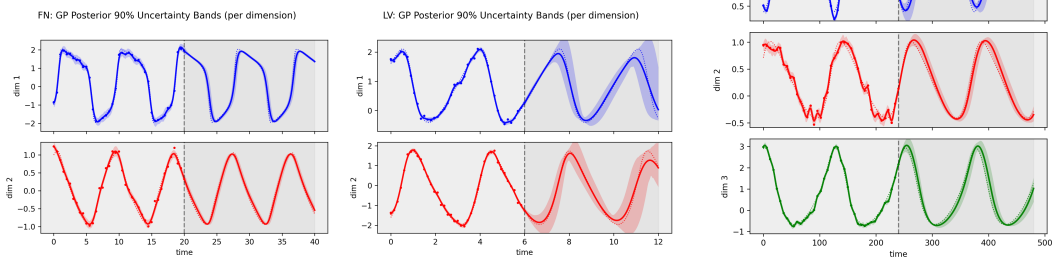
These results show that MAGI-X not only reconstructs trajectories accurately but also learns faithful derivative representations, which are essential for long-term forecasting and system understanding.

3.5 Uncertainty Quantification Calibration and High-dimension Scaling

Gaussian processes provide natural uncertainty quantification. Across three systems, 90% intervals achieve near-nominal coverage (94.6–95.9%), indicating reliable forecasts (Figure 5; details in App.C.3)

Moreover, Magi-X preserves linear runtime scaling, completing a 40-component system in ≈270s. This is orders faster than Neural ODEs, which require longer training even for small systems. This highlights MAGI-X's computational efficiency for high-dimensional inference (details in App.D.3).

	FN	LV	Hes1
Coverage@90 (Width)	85.6% (0.377)	95.9% (1.179)	84.6% (0.446)



(a) FN (per-dimension grid) (b) LV (per-dimension grid) (c) Hes1 (per-dimension grid)

Figure 5: GP-based UQ for FN, LV, and Hes1 (per-dimension grid views). Shown are mean inferred trajectories (solid), ground truth (dotted), observations (points), and 90% credible bands (shaded). The vertical dashed line marks the fit/forecast boundary.

3.6 Real-world Application: Seasonal influenza rolling forecasts

We apply MAGI-X to U.S. influenza surveillance with two coupled signals: weekly hospitalizations (`hosp_US`) and outpatient influenza-like illness (`weighted_ili_US`), obtained from the U.S. Centers for Disease Control and Prevention (CDC) influenza forecasting data portal (Centers for Disease Control and Prevention, 2025). We perform iterative 4-week-ahead rolling forecasts across two years (2023–2025), retraining the model every month on all data observed up to that point. The configuration matches the simulation setup of App. C.2.

As shown in Figure 6, MAGI-X closely tracks the timing and shape of ILI peaks and captures the hospitalization surge/decay patterns without mechanistic priors. Sudden hospitalization spikes (e.g., late 2024) are partially over- or under-shot, but the forecasts remain stable throughout the two-year horizon, demonstrating that the integration-free GP–NN coupling supports rolling, multi-indicator epidemiological forecasting.

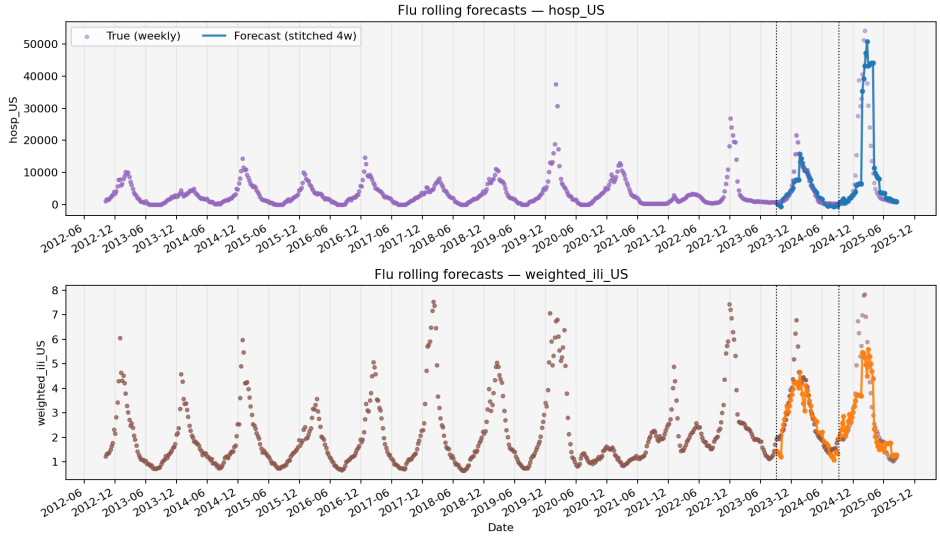


Figure 6: **Iterative 4-week-ahead rolling forecasts for U.S. influenza.** Stitched forecasts from MAGI-X, retrained every 4 weeks on all data observed up to that point, over calendar years 2023–2025. Rows correspond to weekly hospitalizations (`hosp_US`, top) and outpatient influenza-like illness (`weighted_ili_US`, bottom).

4 DISCUSSION

We propose MAGI-X to learn *unknown* ODE systems where the derivative function f^* has no known parametric form. To model this black-box f^*, we use a neural network—an expressive universal approximator with strong practical performance, coupled with a Bayesian GP prior on latent trajectories via a manifold constraint. Across our numerical studies (Sec. 3), MAGI-X not only matches but typically *improves upon* state-of-the-art accuracy, while reducing computational time by bypassing costly numerical integration. In addition, MAGI-X offers natural uncertainty quantification, scales linearly with system dimension, and enables interpretability through visualization of the learned vector field. Finally, the GP state model provides a principled treatment of partially observed systems—an aspect largely neglected in prior work.

Robust optimization via a GP–NN feedback loop. Modeling f^* with a neural network introduces two layers of functional approximation: (i) the latent trajectory $X(t)$ via a manifold-constrained GP, and (ii) the unknown derivative f^* via the network. We address the resulting optimization challenges with a block-wise scheme that alternates between updating the inferred trajectory $x(\mathcal{T})$ and the network parameters θ. Concretely, we apply a small perturbation to $x(\mathcal{T})$ and then update θ to better fit the perturbed path. Training θ against many *probable* $x(\mathcal{T})$ samples mitigates overfitting to sparse, noisy observations $y(\tau)$ and enforces a stable GP–NN feedback. Empirically (Figure 1), the learned f recovers periodic dynamics without imposing a periodic GP prior, and it extrapolates without the divergence that often appears in forecasting when this coupling is absent.

Accuracy, speed, and scalability. Relative to integration-based baselines (e.g., Neural ODE, NPODE), MAGI-X achieves stronger accuracy–time trade-offs: it is frequently more accurate at a fraction of the runtime. The synthetic large-system example confirms that runtime grows approximately linearly with dimension, reinforcing the practicality of MAGI-X for high-dimensional inference and enabling laptop-scale experimentation.

Partial observability and principled uncertainty. Within our Bayesian framework, independent time–GPs per state naturally accommodate asynchronous or missing measurements. This yields competitive reconstruction under partial observation and enables principled latent-trajectory uncertainty quantification that cleanly separates measurement noise from model-induced uncertainty. In practice, coverage and band widths reflect the uncertainty of the terminal state at the fit/forecast split, producing conservative forecasts when that boundary is uncertain and tighter bands when it is well-pinned.

Interpretability: from black box to vector-field insight. Although neural networks are often labeled "black box," MAGI-X estimates the entire vector field, which we can visualize directly (e.g., phase portraits, vector-field arrows, etc). These diagnostic plots provide interpretable evidence that the learned dynamics—not just the trajectory—are faithful. As a follow-up direction, one can recover an explicit parametric form from the learned field (e.g., sparse/symbolic regression seeded by MAGI-X predictions). Such structure discovery would address two classic concerns with powerful nonlinears: (i) limited interpretability and (ii) weaker out-of-domain extrapolation, while offering an end-to-end, data-driven path to mechanistic models without heavy domain priors.

Gray-box calibration via derivative-level discrepancy. A realistic setting is that portions of the ODE are known but imperfect. Prior discrepancy modeling often acts on the *trajectory* level (e.g., Kennedy–O'Hagan). With tools for learning unknown ODEs now in hand, we advocate adapting this framework at the *derivative* level: combine a known parametric form with a learned discrepancy for f, yielding a gray-box model that preserves interpretability where justified and corrects misspecification where needed. We view this as a promising direction for robust extrapolation and principled uncertainty in partially known systems.

LLM Usage Disclosure Portions of the text were assisted by Large Language Models. The authors verified, edited, and take full responsibility for the final content.

Reproducibility Statement. All experimental settings, model configurations, and dataset processing steps are described in the main text and appendix. (Sec. 3, App. C) The code and scripts for reproducing our results will be released on GitHub after the anonymous review process.

REFERENCES

S. Arik, C.-L. Li, J. Yoon, R. Sinha, A. Epshteyn, L. Le, V. Menon, S. Singh, L. Zhang, M. Nikoltchev, Y. Sonthalia, H. Nakhost, E. Kanal, and T. Pfister. Interpretable sequence learning for covid-19 forecasting. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18807–18818. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/d9dbc51dc534921589adf460c85cd824-Paper.pdf.

Y. Bard. Nonlinear parameter estimation. *SIAM Rev.*, 17(4):703–704, Oct. 1975. ISSN 0036-1445. doi: 10.1137/1017088. URL https://doi.org/10.1137/1017088.

B. Batko, M. Gameiro, Y. Hung, W. Kalies, K. Mischaikow, and E. Vieira. Identifying nonlinear dynamics with high confidence from sparse data. *SIAM Journal on Applied Dynamical Systems*, 23(1):383–409, 2024. doi: 10.1137/23M1560252.

M. Benson. Parameter fitting in dynamic models. *Ecological Modelling*, 6(2):97–115, 1979.

W. Bonnaffé and T. Coulson. Fast fitting of neural odes by bayesian neural gradient matching. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=6t2U8zzj1n.

S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113.

B. Calderhead, M. Girolami, and N. D. Lawrence. Accelerating bayesian inference over nonlinear differential equations with gaussian processes. In *Advances in neural information processing systems*, pages 217–224. Citeseer, 2009.

Centers for Disease Control and Prevention. Flusight: Flu forecasting. https://www.cdc.gov/flu-forecasting/index.html, 2025. Accessed: September 2025.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

F. Dondelinger, D. Husmeier, S. Rogers, and M. Filippone. Ode parameter inference using adaptive gradient matching with gaussian processes. In *Artificial intelligence and statistics*, pages 216–228. PMLR, 2013.

R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466, 1961.

J. González, I. Vujačić, and E. Wit. Reproducing kernel hilbert space based estimation of systems of ordinary differential equations. *Pattern Recognition Letters*, 45:26–32, 2014.

M. Heinonen and F. d'Alché Buc. Learning nonparametric differential equations with operator-valued kernels and gradient matching. *arXiv preprint arXiv:1411.5172*, 2014.

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

P. Hennig, M. A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179): 20150142, 2015. ISSN 1364-5021. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2015.0142.

H. Hirata, S. Yoshiura, T. Ohtsuka, Y. Bessho, T. Harada, K. Yoshikawa, and R. Kageyama. Oscillatory expression of the bhlh factor hes1 regulated by a negative feedback loop. *Science*, 298(5594): 840–843, 2002.

D. Karlen. Characterizing the spread of covid-19. *arXiv preprint arXiv:2007.07156*, 2020.

W. O. Kermack and A. G. McKendrick. Contributions to the mathematical theory of epidemics–i. 1927. *Bulletin of mathematical biology*, 53(1-2):33–55, 1991.

Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/pdf/53c47f849d1cd4d21b865caf7d774e07a5c42aa4.pdf.

J. L. Loeppky, J. Sacks, and W. J. Welch. Choosing the sample size of a computer experiment: A practical guide. *Technometrics*, 51(4):366–376, 2009.

A. J. Lotka. The growth of mixed populations: Two species competing for a common food supply. *Journal of the Washington Academy of Sciences*, 22(16/17):461–469, 1932. ISSN 00430439. URL http://www.jstor.org/stable/24530449.

L. Lu, P. Jin, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. ISSN 2522-5839. URL https://www.nature.com/articles/s42256-021-00302-5.

Y.-A. Ma, T. Chen, and E. B. Fox. A complete recipe for stochastic gradient mcmc. In *NIPS*, NIPS'15, page 2917–2925, Cambridge, MA, USA, 2015. MIT Press.

B. Macdonald, C. Higham, and D. Husmeier. Controversy in mechanistic modelling with gaussian processes. In *International conference on machine learning*, pages 1539–1547. PMLR, 2015.

J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962.

M. Niu, S. Rogers, M. Filippone, and D. Husmeier. Fast parameter inference in nonlinear dynamical systems using iterative gradient matching. In *International Conference on Machine Learning*, pages 1699–1707. PMLR, 2016.

J. O. Ramsay, G. Hooker, D. Campbell, and J. Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2006. ISBN 978-0-262-18253-9. URL http://www.gaussianprocess.org/gpml/.

S. Ridderbusch, P. Goulart, and S. Ober-Blöbaum. Learning ode models with qualitative structure using gaussian processes. *arXiv preprint arXiv:2011.05364*, 2020.

C. Rusmassen and C. Williams. *Gaussian process for machine learning*. MIT Press, Cambridge, MA, USA, 2006.

M. Schober, S. Särkkä, and P. Hennig. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, 2019. ISSN 1573-1375. URL https://link.springer.com/article/10.1007/s11222-017-9798-7.

L. Treven, A. Koren, M. Mernik, M. Bosnjak, V. Aravantinos, E. Brochu, R. Gros, H. Yin, D. Vrancic, and M. Munih. Distributional gradient matching for learning uncertain neural dynamics models. In *Advances in Neural Information Processing Systems*, volume 34, pages 16526–16538, 2021.

B. van Domselaar and P. W. Hemker. Nonlinear parameter estimation in initial value problems. *Stichting Mathematisch Centrum. Numerieke Wiskunde*, 75(NW 18), 1975.

J. M. Varah. A spline least squares method for numerical parameter estimation in differential equations. *SIAM Journal on Scientific and Statistical Computing*, 3(1):28–46, 1982.

S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed neural networks. *arXiv preprint arXiv:2111.03794*, 2021. URL https://arxiv.org/abs/2111.03794.

S. Wang, X. Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Y. Wang and D. Barber. Gaussian processes for bayesian estimation in ordinary differential equations. In *International conference on machine learning*, pages 1485–1493. PMLR, 2014.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

P. Wenk, A. Gotovos, S. Bauer, N. S. Gorbach, A. Krause, and J. M. Buhmann. Fast gaussian process based gradient matching for parameter identification in systems of nonlinear odes. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1351–1360. PMLR, 2019.

P. Wenk, G. Abbati, M. A. Osborne, B. Schölkopf, A. Krause, and S. Bauer. Odin: Ode-informed regression for parameter and state inference in time-continuous dynamical systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6364–6371, Apr. 2020. doi: 10.1609/aaai.v34i04.6106. URL https://ojs.aaai.org/index.php/AAAI/article/view/6106.

S. Yang, S. W. Wong, and S. Kou. Inference of dynamic systems from noisy and sparse data via manifold-constrained gaussian processes. *Proceedings of the National Academy of Sciences*, 118 (15), 2021a.

W. Yang, S. Kandula, M. Huynh, S. K. Greene, G. Van Wye, W. Li, H. T. Chan, E. McGibbon, A. Yeung, D. Olson, et al. Estimating the infection-fatality risk of sars-cov-2 in new york city during the spring 2020 pandemic wave: a model-based analysis. *The Lancet Infectious Diseases*, 21(2):203–212, 2021b.

D. Zou, L. Wang, P. Xu, J. Chen, W. Zhang, and Q. Gu. Epidemic model guided machine learning for covid-19 forecasts in the united states. *medRxiv*, 2020. doi: 10.1101/2020.05.24.20111989. URL https://www.medrxiv.org/content/early/2020/05/25/2020.05.24.20111989.

Supplementary Materials

A GAUSSIAN PROCESS

We introduce the scalar-input scalar-output Gaussian process that is concerned in this paper. Following the definition in Rusmassen and Williams (2006), a Gaussian Process is a collection of random variables such that any finite number of which have a joint multivariate Gaussian distribution, denoted by

$$X(t) \sim \mathcal{GP}(\mu, \mathcal{K}_\phi),\ t \in \mathbb{R}, \tag{9}$$

where $\mu : \mathbb{R} \to \mathbb{R}$ is the mean function and $\mathcal{K}_\phi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a positive definite covariance function parameterized by some hyperparameter ϕ. For any finite set of time points $\mathcal{T} = (t_1, \ldots, t_n)$, we have

$$X(\mathcal{T}) \sim \mathcal{N}(\mu(\mathcal{T}), \mathcal{K}_\phi(\mathcal{T}, \mathcal{T})). \tag{10}$$

It is typical that we only have noisy observations of the function values, i.e., we observe $Y(t_i) = X(t_i) + \epsilon_i$ where we assume additive i.i.d. Gaussian random noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. It follows that

$$Y(\mathcal{T}) \sim \mathcal{N}(\mu(\mathcal{T}), \mathcal{K}_\phi(\mathcal{T}, \mathcal{T}) + \sigma^2 I_n). \tag{11}$$

Conditional on observing $Y(\mathcal{T}) = y(\mathcal{T})$, the predictive distribution of X at $\mathcal{T}^* = (t_1^*, \ldots, t_m^*)$ is

$$X(\mathcal{T}^*)|\mathcal{T}, Y(\mathcal{T}) = y(\mathcal{T}) \sim \mathcal{N}(\tilde{\mu}(\mathcal{T}^*), \tilde{\text{cov}}(\mathcal{T}^*)), \tag{12}$$

where

$$\begin{aligned} \tilde{\mu}(\mathcal{T}^*) &= \mu(\mathcal{T}^*) + \mathcal{K}_\phi(\mathcal{T}^*, \mathcal{T})(\mathcal{K}_\phi(\mathcal{T}, \mathcal{T}) + \sigma^2 I_n)^{-1}(y(\mathcal{T}) - \mu(\mathcal{T})), \\ \tilde{\text{cov}}(\mathcal{T}^*) &= \mathcal{K}_\phi(\mathcal{T}^*, \mathcal{T}^*) - \mathcal{K}_\phi(\mathcal{T}^*, \mathcal{T})(\mathcal{K}_\phi(\mathcal{T}, \mathcal{T}) + \sigma^2 I_n)^{-1}\mathcal{K}_\phi(\mathcal{T}, \mathcal{T}^*), \end{aligned} \tag{13}$$

can be derived using the property of conditional multivariate Gaussian distribution.

Derivative of Gaussian process. Let us now derive the first order derivative $\dot{X}(t)$ with respect to the input t. Since differentiation is a linear operator, the derivative of a Gaussian process is again a Gaussian process with some mean $\dot{\mu}$ (see Appendix of Wenk et al. (2019)), and the joint distribution of $X(t)$ and $\dot{X}(t)$ is

$$\begin{bmatrix} X(t) \\ \dot{X}(t) \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} \mu \\ \dot{\mu} \end{bmatrix}, \begin{bmatrix} \mathcal{K}_\phi & \mathcal{K}'_\phi \\ {}'\mathcal{K}_\phi & \mathcal{K}''_\phi \end{bmatrix} \right), \tag{14}$$

where ${}'\mathcal{K}_\phi = \frac{\partial}{\partial s}\mathcal{K}_\phi(s,t)$, $\mathcal{K}'_\phi = \frac{\partial}{\partial t}\mathcal{K}_\phi(s,t)$, and $\mathcal{K}''_\phi = \frac{\partial}{\partial s \partial t}\mathcal{K}_\phi(s,t)$. Conditional on observing $X(\mathcal{T}) = x(\mathcal{T})$, the distribution of $\dot{X}(\mathcal{T})$ is

$$\dot{X}(\mathcal{T})|X(\mathcal{T}) = x(\mathcal{T}) \sim \mathcal{N}(\dot{\tilde{\mu}}(\mathcal{T}), \dot{\tilde{\text{cov}}}(\mathcal{T})), \tag{15}$$

where

$$\begin{aligned} \dot{\tilde{\mu}}(\mathcal{T}) &= \dot{\mu}(\mathcal{T}) + {}'\mathcal{K}_\phi(\mathcal{T}, \mathcal{T})\mathcal{K}_\phi(\mathcal{T}, \mathcal{T})^{-1}(x(\mathcal{T}) - \mu(\mathcal{T})), \\ \dot{\tilde{\text{cov}}}(\mathcal{T}) &= \mathcal{K}''_\phi(\mathcal{T}, \mathcal{T}) - {}'\mathcal{K}_\phi(\mathcal{T}, \mathcal{T})\mathcal{K}_\phi(\mathcal{T}, \mathcal{T})^{-1}\mathcal{K}'_\phi(\mathcal{T}, \mathcal{T}), \end{aligned} \tag{16}$$

are derived again by the property of conditional multivariate Gaussian distribution.

Hyperparameter Tuning. We now discuss the tuning of the hyperparameters ϕ. Noted from previous studies, the performance of the Gaussian process based gradient constrained methods rely heavily on the quality of the hyperparameters ϕ. Following Wenk et al. (2019; 2020); Yang et al. (2021a), we employ the empirical Bayes method to choose the hyperparameters ϕ and the noise variance σ^2 that maximizes the marginal likelihood of the observations $y(\tau)$ defined in equation 11, that is to solve

$$\arg\max_{\phi, \sigma^2} p_{\mathcal{N}}\Big(y(\tau); \mu(\mathcal{T}), \mathcal{K}_\phi(\mathcal{T}, \mathcal{T}) + \sigma^2 I_n\Big) \tag{17}$$

where $p_{\mathcal{N}}(\cdot; \mu, \Sigma)$ is the p.d.f. of multivariate Gaussian distribution with mean μ and variance Σ. If $\mu(\mathcal{T})$ is also not known apriori, we usually consider the prior mean function to be a constant function, i.e. $\mu(\cdot) = c$ for some unknown constant $c \in \mathbb{R}$. This is employed in our implementation, and we find c by optimizing over c along with ϕ and σ^2 in equation 17. In the case where $\mu(\cdot) = c$ is a constant function, then $\dot{\mu}(\cdot) = 0$ in equation 14.

Matérn covariance function. In this paper, we use the Matérn kernel with degree of freedom 2.01. The Matérn covariance function between any two points t_1 and t_2 with euclidean distance $d = \|t_1 - t_2\|_2$ is

$$\mathcal{K}_\nu(t_1, t_2) = \mathcal{K}_\nu(d) = \omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{d}{\rho} \right)^\nu B_\nu \left(\sqrt{2\nu} \frac{d}{\rho} \right), \tag{18}$$

where Γ is the gamma function, B_ν is the modified Bessel function of the second kind, ν is the associated degree of freedom, ω is the variance parameter, and ρ is the lengthscale parameter. The hyperparameters are $\phi = \{\omega, \rho\}$. Note that the modified Bessel function of the second kind satisfies the following recurrence relations:

$$\begin{aligned} \frac{-2\nu}{r} B_\nu(r) &= B_{\nu-1}(r) - B_{\nu+1}(r), \\ B'_\nu(r) &= -\frac{B_{\nu-1}(r) + B_{\nu+1}(r)}{2}, \end{aligned} \tag{19}$$

and the following limit conditions:

$$\begin{aligned} \lim_{r \to 0} B_\nu(r) &= \infty, \\ \lim_{r \to 0} r^\nu B_\nu(r) &= \frac{\Gamma(\nu)}{2^{1-\nu}}. \end{aligned} \tag{20}$$

Thus, it follows that

$$\lim_{d \to 0} K_\nu(d) = \omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \times \frac{\Gamma(\nu)}{2^{1-\nu}} = \omega^2. \tag{21}$$

Let us now compute the partial derivative of Matérn covariance function with respect to $r = \sqrt{2\nu} d / \rho$,

$$\begin{aligned} \frac{\partial \mathcal{K}_\nu}{\partial r} &= \omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left\{ \nu r^{\nu-1} B_\nu(r) + r^\nu B'_\nu(r) \right\} \\ &= \left\{ \omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} r^\nu B_\nu(r) \right\} \left\{ \frac{\nu}{r} + \frac{B'_\nu(r)}{B_\nu(r)} \right\} \\ &= \mathcal{K}_\nu(d) \left\{ \frac{\nu}{r} + \frac{B'_\nu(r)}{B_\nu(r)} \right\}, \end{aligned} \tag{22}$$

with $\lim_{r \to 0} \partial \mathcal{K}_\nu / \partial r = 0$ (see App. B.3). Given that we only consider one-dimensional inputs, the euclidean distance function can be simplified to be $d = |t_1 - t_2|$, and thus we have

$$\begin{aligned} \frac{\partial d}{\partial t_1} &= \frac{t_1 - t_2}{|t_1 - t_2|} \text{ if } t_1 \neq t_2 \text{ and } 0 \text{ if } t_1 = t_2, \\ \frac{\partial d}{\partial t_2} &= -\frac{t_1 - t_2}{|t_1 - t_2|} \text{ if } t_1 \neq t_2 \text{ and } 0 \text{ if } t_1 = t_2. \end{aligned} \tag{23}$$

One can recognize that $\partial d / \partial t_2 = -\partial d / \partial t_1$. By applying the chain rule, we can compute the first order partial derivative with respect to t_1 by

$$\frac{\partial \mathcal{K}_\nu}{\partial t_1} = \frac{\partial \mathcal{K}_\nu}{\partial r} \frac{\partial r}{\partial d} \frac{\partial d}{\partial t_1}, \tag{24}$$

where $\partial r / \partial d = \sqrt{2\nu} / \rho$, and we can then compute $\partial \mathcal{K}_\nu / \partial t_2 = -\partial \mathcal{K}_\nu / \partial t_1$. Next, let us compute the second order partial derivative with respect to both t_1 and t_2, that is

$$\frac{\partial^2 \mathcal{K}_\nu}{\partial t_1 \partial t_2} = \frac{\partial}{\partial t_1} \left(\frac{\partial \mathcal{K}_\nu}{\partial d} \frac{\partial d}{\partial t_2} \right) = \frac{\partial^2 \mathcal{K}_\nu}{\partial d^2} \frac{\partial d}{\partial t_1} \frac{\partial d}{\partial t_2} + \frac{\partial \mathcal{K}_\nu}{\partial d} \frac{\partial^2 d}{\partial t_1 \partial t_2} = -\frac{\partial^2 \mathcal{K}_\nu}{\partial d^2}, \tag{25}$$

by the fact that $\frac{\partial d}{\partial t_1} \frac{\partial d}{\partial t_2} = -(\frac{\partial d}{\partial t_1})^2 = -1$ and $\frac{\partial^2 d}{\partial t_1 \partial t_2} = 0$. This leave us to compute $\partial^2 \mathcal{K}_\nu / \partial d^2$. Similarly, by using chain rule, we have

$$\frac{\partial^2 \mathcal{K}_\nu}{\partial d^2} = \frac{\partial}{\partial d} \left(\frac{\partial \mathcal{K}_\nu}{\partial r} \frac{\partial r}{\partial d} \right) = \frac{\partial^2 \mathcal{K}_\nu}{\partial r^2} \frac{\partial r}{\partial d} \frac{\partial r}{\partial d} + \frac{\partial \mathcal{K}_\nu}{\partial r} \frac{\partial^2 r}{\partial d^2} = \frac{\partial^2 \mathcal{K}_\nu}{\partial r^2} \left(\frac{\partial r}{\partial d} \right)^2, \tag{26}$$

where $\partial^2 r/\partial d^2 = 0$ from $r = \sqrt{2\nu}d/\rho$ and

$$\begin{aligned}\frac{\partial^2 \mathcal{K}_\nu}{\partial r^2} &= \omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left\{ \nu(\nu-1) r^{\nu-2} B_\nu(r) + 2\nu r^{\nu-1} B'_\nu(r) + r^\nu B''_\nu(r) \right\} \\ &= \left\{ \omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} r^\nu B_\nu(r) \right\} \left\{ \frac{\nu(\nu-1)}{r^2} + \frac{2\nu}{r} \frac{B'_\nu(r)}{B_\nu(r)} + \frac{B''_\nu(r)}{B_\nu(r)} \right\} \\ &= \mathcal{K}_\nu(d) \left\{ \frac{\nu(\nu-1)}{r^2} + \frac{2\nu}{r} \frac{B'_\nu(r)}{B_\nu(r)} + \frac{B''_\nu(r)}{B_\nu(r)} \right\},\end{aligned} \tag{27}$$

with $\lim_{r\to 0} \partial^2 \mathcal{K}_\nu / \partial r^2 = -\omega^2 \frac{1}{2(\nu-1)}$ (see App. B.3).

B Model Objective and Proofs

B.1 Log Posterior

In this supplementary material, we omit the superscripts on ODE derivative function f for simplicity. Following the results in App. A, we have

$$X_d(\mathcal{T}) \sim \mathcal{N}\Big(\mu_d(\mathcal{T}), C_d\Big), \tag{28}$$

where $C_d = \mathcal{K}_d(\mathcal{T}, \mathcal{T})$, and

$$\dot{X}_d(\mathcal{T}) | X_d(\mathcal{T}) = x_d(\mathcal{T}) \sim \mathcal{N}\Big(\dot{\tilde{\mu}}_d(\mathcal{T}), K_d\Big), \tag{29}$$

where

$$\begin{aligned}\dot{\tilde{\mu}}_d(\mathcal{T}) &= \dot{\mu}_d(\mathcal{T}) + {}'\mathcal{K}_d(\mathcal{T}, \mathcal{T}) \mathcal{K}_d(\mathcal{T}, \mathcal{T})^{-1} (x_d(\mathcal{T}) - \mu_d(\mathcal{T})), \\ K_d &= \mathcal{K}''_d(\mathcal{T}, \mathcal{T}) - {}'\mathcal{K}_d(\mathcal{T}, \mathcal{T}) \mathcal{K}_d(\mathcal{T}, \mathcal{T})^{-1} \mathcal{K}'_d(\mathcal{T}, \mathcal{T}),\end{aligned} \tag{30}$$

follows from equation 16. Thus,

$$\begin{aligned}&p(W_\mathcal{T} = 0 | X(\mathcal{T}) = x(\mathcal{T}), \Theta = \theta) \\ &= \prod_{d=1}^{D} p\Big(\dot{X}_d(\mathcal{T}) = \{f(x(\mathcal{T}), \theta, t_\mathcal{T})\}_d | X(\mathcal{T}) = x(\mathcal{T}), \Theta = \theta \Big) \\ &= \prod_{d=1}^{D} p_\mathcal{N}\Big(\{f(x(\mathcal{T}), \theta, t_\mathcal{T})\}_d; \dot{\tilde{\mu}}_d(\mathcal{T}), K_d \Big),\end{aligned} \tag{31}$$

where $p_\mathcal{N}(\cdot; \mu, \Sigma)$ is the p.d.f. of multivariate Gaussian distribution with mean μ and variance Σ. Thus, the log posterior distribution function is

$$\begin{aligned}&\log p\{\Theta = \theta, X(\mathcal{T}) = x(\mathcal{T}) | W_\mathcal{T} = 0, Y(\tau) = y(\tau)\} \\ &= \text{Const.} + \log \pi_\Theta(\theta) - \frac{1}{2} \sum_{d=1}^{D} \Big\{ \log |C_d| + \| x_d(\mathcal{T}) - \mu_d(\mathcal{T}) \|^2_{C_d^{-1}} + \\ &N_d \log(\sigma_d^2) + \| y_d(\tau_d) - x_d(\tau_d) \|^2_{\sigma_d^{-2} I_{N_d}} + \log |K_d| + \| \{f(x(\mathcal{T}), \theta, t_\mathcal{T})\}_d - \dot{\tilde{\mu}}_d(\mathcal{T}) \|_{K_d^{-1}} \Big\}\end{aligned} \tag{32}$$

where $\|v\|^2_A = v^T A v$ and $|A|$ is the determinant of A.

B.2 Discretization error for the manifold constraint

Let $T = \{t_1 < \cdots < t_n\} \subset [0, T]$ be a grid with mesh size $h = \max_i (t_{i+1} - t_i)$. For $d \in \{1, \ldots, D\}$ define

$$s_d(t) \;=\; \dot{X}_d(t) - f_d\big(X(t), \theta, t\big), \qquad W \;=\; \max_d \sup_{t \in [0,T]} |s_d(t)|, \qquad W_T \;=\; \max_d \max_{t_i \in T} |s_d(t_i)|.$$

Clearly $0 \le W - W_T$ since $T \subset [0, T]$.

Theorem B.1 (Linear-rate bound)**.** *Assume $s_d \in C^1[0,T]$ and $\sup_{t\in[0,T]} |s_d'(t)| \le M_{1,d}$ for each d. Then*

$$0 \;\le\; W - W_T \;\le\; \frac{h}{2}\, M_1, \qquad M_1 := \max_d M_{1,d}.$$

Proof. Fix d and an interval $[a,b] = [t_i, t_{i+1}]$ with $b-a \le h$. Let $t^\star \in [a,b]$ attain $\max_{t\in[a,b]} |s_d(t)|$ (exists by continuity). Choose the nearer endpoint $e \in \{a,b\}$ so that $|t^\star - e| \le (b-a)/2$. Then by the reverse triangle inequality,

$$|s_d(t^\star)| - \max\{|s_d(a)|, |s_d(b)|\} \;\le\; |s_d(t^\star)| - |s_d(e)| \;\le\; |s_d(t^\star) - s_d(e)|.$$

By the mean value theorem and the bound on $|s_d'|$, $|s_d(t^\star) - s_d(e)| \le M_{1,d}\, |t^\star - e| \le \frac{1}{2} M_{1,d}(b-a) \le \frac{h}{2} M_{1,d}$. Hence

$$\sup_{t\in[t_i,t_{i+1}]} |s_d(t)| \;\le\; \max\{|s_d(t_i)|, |s_d(t_{i+1})|\} + \tfrac{h}{2} M_{1,d}.$$

Maximizing over intervals and then over d yields the claim. □

B.3 Proof for the limit of Matérn covariance function's derivative at 0

Claim 1: $\lim_{r\to 0} \frac{\partial \mathcal{K}_\nu}{\partial r} = 0$.

Proof. Follow from equation 22 and apply the recurrence relations equation 19 and the limit conditions equation 20 of the modified Bessel function of the second kinds, we have

$$\begin{aligned}
&\lim_{r\to 0} \frac{\partial \mathcal{K}_\nu}{\partial r} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \left\{ \frac{\nu}{r} r^\nu B_\nu(r) + r^\nu B_\nu'(r) \right\} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \left\{ \frac{\nu}{r} r^\nu B_\nu(r) + r^\nu \left(- \frac{B_{\nu-1}(r) + B_{\nu+1}(r)}{2} \right) \right\} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \left\{ \frac{\nu r^\nu B_\nu(r) - \frac{1}{2} r^{\nu+1} B_{\nu+1}(r)}{r} - \frac{1}{2} r^\nu B_{\nu-1}(r) \right\} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \frac{\nu r^\nu B_\nu(r) - \frac{1}{2} r^{\nu+1} B_{\nu+1}(r)}{r} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \frac{\nu r^\nu \{ -\frac{r}{2\nu} (B_{\nu-1}(r) - B_{\nu+1}(r)) \} - \frac{1}{2} r^{\nu+1} B_{\nu+1}(r)}{r} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \frac{-\frac{1}{2} r^{\nu+1} B_{\nu-1}(r)}{r} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} -\frac{1}{2} r^\nu B_{\nu-1}(r) \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} r \left(- \frac{1}{2} r^{\nu-1} B_{\nu-1}(r) \right) \\
=&0
\end{aligned} \tag{33}$$

□

Claim 2: $\lim_{r\to 0} \frac{\partial^2 \mathcal{K}_\nu}{\partial r^2} = -\omega^2 \frac{1}{2(\nu-1)}$.

Proof. Recursively applying the recurrence relations equation 19, we can compute that

$$B''(r) = \frac{1}{4} \left(B_{\nu-2}(r) + 2B_\nu(r) + B_{\nu+2}(r) \right). \tag{34}$$

Thus, applying the recurrence relations equation 19 and the limit conditions equation 20 to equation 27 as $r \to 0$, we have

$$
\begin{aligned}
&\lim_{r\to 0} \frac{\partial^2 \mathcal{K}_\nu}{\partial r^2} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \left\{ \frac{\nu(\nu-1)}{r^2} r^\nu B_\nu(r) + \frac{2\nu}{r} r^\nu B'(\nu) + r^\nu B''_\nu(r) \right\} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \left\{ \frac{\nu(\nu-1)}{r^2} r^\nu B_\nu(r) + \frac{2\nu}{r} r^\nu \left(- \frac{B_{\nu-1}(r) + B_{\nu+1}(r)}{2} \right) + \right. \\
&\left. r^\nu \frac{1}{4} \left(B_{\nu-2}(r) + 2B_\nu(r) + B_{\nu+2}(r) \right) \right\} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \left\{ \frac{\nu(\nu-1)}{r^2} r^\nu B_\nu(r) - \nu r^{\nu-1} B_{\nu-1}(r) - \frac{\nu}{r^2} r^{\nu+1} B_{\nu+1}(r) + \right. \\
&\left. \frac{r^2}{4} r^{\nu-2} B_{\nu-2}(r) + \frac{1}{2} r^\nu B_\nu(r) + \frac{1}{r^2} \frac{1}{4} r^{\nu+2} B_{\nu+2}(r) \right\} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \left\{ - r^{\nu-1} B_{\nu-1}(r) + \left(- (\nu-1) r^{\nu-1} B_{\nu-1}(r) + \frac{1}{2} r^\nu B_\nu(r) \right) + \right. \\
&\left. \frac{\nu(\nu-1) r^\nu B_\nu(r) - \nu r^{\nu+1} B_{\nu+1}(r) + \frac{1}{4} r^{\nu+2} B_{\nu+2}(r)}{r^2} \right\} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \lim_{r\to 0} \left\{ - r^{\nu-1} B_{\nu-1}(r) \right\} \\
=&\omega^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left\{ - \frac{\Gamma(\nu-1)}{2^{1-(\nu-1)}} \right\} \\
=&- \omega^2 \frac{1}{2(\nu-1)}
\end{aligned}
\tag{35}
$$

where

$$
\lim_{r\to 0} \left(- (\nu-1) r^{\nu-1} B_{\nu-1}(r) + \frac{1}{2} r^\nu B_\nu(r) \right) = -(\nu-1) \frac{\Gamma(\nu-1)}{2^{1-(\nu-1)}} + \frac{1}{2} \cdot \frac{\Gamma(\nu)}{2^{1-\nu}} = 0 \tag{36}
$$

and

$$
\begin{aligned}
&\lim_{r\to 0} \frac{\nu(\nu-1) r^\nu B_\nu(r) - \nu r^{\nu+1} B_{\nu+1}(r) + \frac{1}{4} r^{\nu+2} B_{\nu+2}(r)}{r^2} \\
=&\lim_{r\to 0} \frac{\nu(\nu-1) r^\nu \{ -\frac{r}{2\nu} (B_{\nu-1}(r) - B_{\nu+1}(r)) \} - \nu r^{\nu+1} B_{\nu+1}(r) + \frac{1}{4} r^{\nu+2} B_{\nu+2}(r)}{r^2} \\
=&\lim_{r\to 0} \frac{-\frac{\nu-1}{2} r^{\nu+1} B_{\nu-1}(r) + \frac{\nu-1}{2} r^{\nu+1} B_{\nu+1}(r) - \nu r^{\nu+1} B_{\nu+1}(r) + \frac{1}{4} r^{\nu+2} B_{\nu+2}(r)}{r^2} \\
=&\lim_{r\to 0} \frac{-\frac{\nu-1}{2} r^{\nu+1} \{ -\frac{r}{2(\nu-1)} (B_{\nu-2}(r) - B_\nu(r)) \} - \frac{\nu+1}{2} r^{\nu+1} B_{\nu+1}(r) + \frac{1}{4} r^{\nu+2} B_{\nu+2}(r)}{r^2} \\
=&\lim_{r\to 0} \frac{-\frac{1}{4} r^{\nu+2} B_\nu(r) - \frac{\nu+1}{2} r^{\nu+1} B_{\nu+1}(r) + \frac{1}{4} r^{\nu+2} B_{\nu+2}(r)}{r^2} \\
=&\lim_{r\to 0} \frac{-\frac{\nu+1}{2} r^{\nu+1} B_{\nu+1}(r) - \frac{1}{4} r^{\nu+2} (B_\nu(r) - B_{\nu+2}(r))}{r^2} \\
=&\lim_{r\to 0} \frac{-\frac{\nu+1}{2} r^{\nu+1} B_{\nu+1}(r) - \frac{1}{4} r^{\nu+2} \{ -\frac{2(\nu+1)}{r} B_{\nu+1}(r) \}}{r^2} \\
=&\lim_{r\to 0} \frac{-\frac{\nu+1}{2} r^{\nu+1} B_{\nu+1}(r) + \frac{\nu+1}{2} r^{\nu+1} B_{\nu+1}(r)}{r^2} \\
=&\lim_{r\to 0} \frac{0}{r^2} \\
=&0
\end{aligned}
\tag{37}
$$

and the last equality follows from the L'Hôspital's Rule.

□

C Implementation Details

In this section discuss our experiment settings(App. C.1 – C.3), and the implementation details that we employ in implementation to avoid numerical instability(App. C.4 – C.5).

C.1 Benchmark Real World Systems

In this paper, we consider the following three benchmark real world systems:

- The FitzHugh-Nagumo (FN) system was introduced by FitzHugh (1961) and Nagumo et al. (1962) for modeling the activation of an excitable system such as neuron. It is a two-component system determined by the following ODEs,

$$\begin{cases} \dot{x}_1 = c(x_1 - x_1^3/3 + x_2) \\ \dot{x}_2 = -(x_1 - a + bx_2)/c \end{cases} \tag{38}$$

  where $a = 0.2$, $b = 0.2$, $c = 3$, and $x(0) = (-1, 1)$. The groundtruth trajectory from $t = 0$ to $t = 40$ obtained via numerical integration is presented in the left panel of Figure 7.

- The Lotka-Volterra (LV) system was introduced by Lotka (1932) for modeling the dynamics of the predator-prey interaction. It is a two-component system determined by the following ODEs,

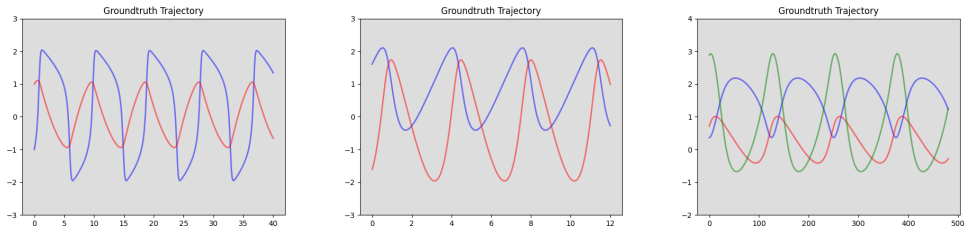
$$\begin{cases} \dot{x}_1 = ax_1 - bx_1x_2 \\ \dot{x}_2 = cx_1x_2 - dx_2 \end{cases} \tag{39}$$

  where $a = 1.5$, $b = 1$, $c = 1$, $d = 3$, and $x(0) = (5, 0.2)$. Given that both x_1 and x_2 are always strictly positive, we consider the log-transformation of the system in this paper. The groundtruth trajectory from $t = 0$ to $t = 12$ obtained via numerical integration is presented in the middle panel of Figure 7.

- The Hes1 system was introduced by Hirata et al. (2002) for modeling the oscillation dynamic of Hes1 protein level (x_1) and Hes1 mRNA level (x_2) under the influence of a Hes1-interacting factor (x_3). It is a three-component system determined by the following ODEs,

$$\begin{cases} \dot{x}_1 = -ax_1x_3 + bx_2 - cx_1 \\ \dot{x}_2 = -dx_2 + e/(1 + x_1^2) \\ \dot{x}_3 = -ax_1x_3 + f/(1 + x_1^2) - gx_3 \end{cases} \tag{40}$$

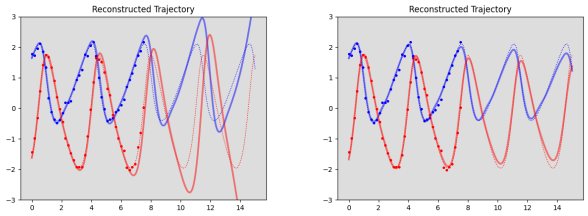
  where $a = 0.022$, $b = 0.3$, $c = 0.031$, $d = 0.028$, $e = 0.5$, $f = 20$, $g = 0.3$, and $x(0) = (1.438575, 2.037488, 17.90385)$. Similarly, given that x_1, x_2, and x_3 are always strictly positive, we consider the log-transformation of the system in this paper. The groundtruth trajectory from $t = 0$ to $t = 480$ obtained via numerical integration is presented in the right panel of Figure 7.



(a) FN (b) LV (log-transformed) (c) Hes1 (log-transformed)

Figure 7: Groundtruth trajectory obtained using numerical integration.

However, we do observe some large standard deviations in the trajectory RMSEs from Table 1, e.g., the trajectory RMSE of x_1 for the Hes1 system. This large standard deviation is due to one divergence case where the small errors accumulate in the trajectory propagation via numerical integration in the forecasting phase, and eventually push the trajectory to unseen domain where we do not have data for



(a) Seed 1 (b) Seed 2

Figure 8: Comparison of the reconstructed trajectory (solid line) to the groundtruth trajectory (dotted lines) after applying MAGI-X on the same set of 41-point noisy data (circles) from the LV system but with two different initialized random seed for the optimization procedure.

the neural network to learn the correct behavior (left panel of Figure 8). However, using a different initialized random seed for the optimization solves the problem (right panel of Figure 8). Thus, in real world usage, we can actually restart MAGI-X with multiple random seeds and select the best one to avoid the possible divergence, and this could still result in faster runtime than NPODE and Neural ODE given the computational saving achieved by MAGI-X.

Nevertheless, improvements to the optimization procedure are still encouraged. One promising solution is to provide a more variety of trajectories $x(\mathcal{T})$ but are still within some credible interval of the true trajectory for training θ, so the model has data for learning about auto-correction once the trajectory slightly deviates. Thus, one future direction is to replace SGD for the update of $x(\mathcal{T})$ by the stochastic gradient Langevin dynamics (Welling and Teh, 2011) or stochastic gradient Markov Chain Monte Carlo (Ma et al., 2015) that are both stochastic gradient based samplers with carefully injected noise.

C.2 MAGI-X Settings

First, for the discretization $\mathcal{T}$, since $y(\tau)$ come from the fitting window, we evaluate on the first 161 equally spaced times $\mathcal{T} = \{t_1, \ldots, t_{161}\}$, which is four times denser than the observation grid τ. This also makes it straightforward to compute the RMSE of the *inferred* trajectory. As a rule of thumb, we recommend using a grid at least $4\times$ finer than the number of available observations. For the GP prior, we adopt a Matérn kernel with smoothness $\nu = 2.01$ so the process is exactly twice differentiable; in our setting this is more robust than an RBF kernel, whose derivative covariance tends to be overly smooth. The neural network uses ReLU activation with a single hidden layer of 512 units, motivated by links between wide networks and kernel methods (Wang et al., 2022); comparisons of alternative architectures are provided in the supplement. We run MAGI-X for 2,500 iterations.

C.3 Details of GP-based Uncertainty Quantification

Setup. For each component $d \in \{1, \ldots, D\}$ we place an independent GP prior on the latent trajectory $x_d(\cdot)$ with Matérn kernel and SKI (KISS-GP) interpolation on the 1D inducing grid $\mathcal{G} = \{\tau_1, \ldots, \tau_T\}$. Let t_{fit} be the last observation time and $\mathcal{G}_{\text{fit}} = \{\tau_j \le t_{\text{fit}}\}$.

Posterior sampling on the fit grid. Compute the per-dimension GP posterior $\mathcal{N}(\boldsymbol{\mu}_d, \Sigma_d)$ at $\mathcal{G}_{\text{fit}}$ using the standard formulas on the *actual* SKI grid. Draw N samples $\mathbf{z}_d^{(n)} \sim \mathcal{N}(\boldsymbol{\mu}_d, \Sigma_d)$ and stack $Z_{\text{fit}}^{(n)} \in \mathbb{R}^{T_{\text{fit}} \times D}$.

Deterministic propagation. For each n, set $x_0^{(n)} = Z_{\text{fit}}^{(n)}[T_{\text{fit}}, :]$ and propagate on a uniform forecast grid $\mathcal{G}_{\text{fct}} = \{\tau_{T_{\text{fit}}} < \cdots < \tau_{T_{\text{full}}}\}$ with horizon chosen as a fraction β of the fit window (default $\beta = 1$):

$$\dot{x}(t) = f_{\hat{\theta}}(x(t)), \qquad x(t_{\text{fit}}) = x_0^{(n)}.$$

Forecast uncertainty thus comes *only* from the terminal state; we do not inject process noise or integrate over $\hat{\theta}$ (dropout disabled).

Pointwise bands. For $\alpha = 0.10$, define empirical $(1-\alpha)$ credible bands from the ensemble $\{\tilde{X}^{(n)}\}_{n=1}^{N}$:

$$\ell_{j,d} = \mathrm{Quantile}_{\alpha/2}\{\tilde{X}_{j,d}^{(n)}\}, \quad u_{j,d} = \mathrm{Quantile}_{1-\alpha/2}\{\tilde{X}_{j,d}^{(n)}\}, \quad \bar{x}_{j,d} = \tfrac{1}{N}\sum_{n} \tilde{X}_{j,d}^{(n)}.$$

Bands target the latent $x(t)$, not observation intervals.

Calibration metrics. For a region $\mathcal{R} \subset \{1{:}T_{\mathrm{full}}\}$ (fit/forecast) and reference trajectory $x^{\star}$:

$$\mathrm{Cov}(\mathcal{R}) = \frac{1}{|\mathcal{R}|D}\sum_{j\in\mathcal{R}}\sum_{d=1}^{D}\mathbf{1}\{x_d^{\star}(t_j) \in [\ell_{j,d}, u_{j,d}]\}, \quad \mathrm{Width}(\mathcal{R}) = \frac{1}{|\mathcal{R}|D}\sum_{j\in\mathcal{R}}\sum_{d=1}^{D}(u_{j,d}-\ell_{j,d}),$$

$$\mathrm{Width}_{\mathrm{norm}}(\mathcal{R}) = \frac{1}{D}\sum_{d=1}^{D}\frac{\frac{1}{|\mathcal{R}|}\sum_{j\in\mathcal{R}}(u_{j,d}-\ell_{j,d})}{\max_{j\in\mathcal{R}} x_d^{\star}(t_j) - \min_{j\in\mathcal{R}} x_d^{\star}(t_j)}.$$

Implementation notes. Use the model's SKI grid $\mathcal{G}$; add jitter and, if needed, clip small eigenvalues for SPD; propagate with a deterministic ODE solver (e.g., `odeint`) under $f_{\hat{\theta}}$.

C.4 Time standardization

Given that the FN, LV, and Hes1 examples share similar magnitude in their component values but have very different time ranges (Figure 7), some time standardization might help improve the robustness of the algorithm. Though different time unit would give theoretically equivalent system, we do observe that NPODE of Heinonen et al. (2018) yields very bad performance on the Hes1 example, but such problem does not exist for the FN and LV systems (time range in the scale of 10s). Same phenomenon is also observed in MAGI-X *without* the time standardization. Given that both NPODE and MAGI-X rely on gradient update for parameters optimization, the performance is sensitive to the learning rate. Thus, the poor performance on the Hes1 example suggests that the learning rate chosen in NPODE and MAGI-X is robust for system with time scale of 10s, but it could be too large for system with time scales of 100s if the component values have similar magnitude. This leads to the need of time standardization. From simulation studies on the three benchmark problems with the specified learning rate, we suggest the following standardization scheme: standardize the data such that the distance between any two nearby time points is 0.05. Given that $\mathcal{T}$ is the 161 equal spaced-out time points from fitting phase, we are essentially standardizing the fitting phase time range to 8 for all systems. What we propose now is an engineering solution, and a better standardization procedure will be investigated in future works.

C.5 Cholesky decomposition

Optimizing over $x(\mathcal{T})$ directly in the objective function of MAGI-X is usually not numerically preferred as the entries of $x_d(\mathcal{T})$ are supposed to be correlated by the GP prior. Now consider $U(\mathcal{T}) = (U_1(\mathcal{T}), \ldots, U_D(\mathcal{T}))$ where

$$U_d(\mathcal{T}) = L_{C_d}^{-1}(X_d(\mathcal{T}) - \mu_d(\mathcal{T})) \quad \Leftrightarrow \quad X_d(\mathcal{T}) = \mu_d(\mathcal{T}) + L_{C_d}U_d(\mathcal{T}). \tag{41}$$

L_{C_d} is the Cholesky decomposition of $C_d = \mathcal{K}_d(\mathcal{T},\mathcal{T})$, i.e., $C_d = L_{C_d}L_{C_d}^T$. We can see that $U_d \sim \mathcal{N}(0, I_{|\mathcal{T}|})$, where all the entries are independent. We make the change of variable from $X(\mathcal{T})$ to $U(\mathcal{T})$ in the objective function, and optimize over $u(\mathcal{T}) = (u_1(\mathcal{T}), \ldots, u_D(\mathcal{T})$ where $u_d(\mathcal{T}) = L_{C_d}^{-1}(x_d(\mathcal{T}) - \mu_d(\mathcal{T}))$ in the actual implementation. This leads to more robust performance when gradient update is employed.

D Additional Figures and Ablations

D.1 Additional NPODE Iterations

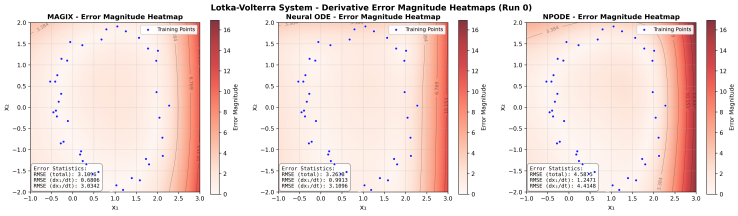
Table 4: RMSE (mean ± std) over 100 runs with full observations. Runtime (RT, seconds) shown per dataset block. **NPODE: 2000 iterations**. Lower is better.

		FN			LV			Hes1			
Phase	Model	x_1	x_2	**RT (s)**	x_1	x_2	**RT (s)**	x_1	x_2	x_3	**RT (s)**
Fitting	NPODE	0.19±0.02	0.05±0.01	135.57±4.03	0.06±0.02	0.07±0.02	137.31±4.72	2.71±1.90	2.55±1.79	2.77±1.70	328.11±1.53
	Neural ODE (512)	1.14±0.37	0.52±0.20	424.05±22.54	0.29±0.03	0.37±0.05	332.22±13.02	0.61±0.12	0.54±0.08	1.23±0.06	1770.79±490.00
	MAGI–X (512)	0.11±0.02	0.05±0.01	23.87±0.75	0.05±0.01	0.05±0.02	23.56±0.52	0.09±0.06	0.06±0.04	0.12±0.10	93.19±1.76
Forecasting	NPODE	0.20±0.07	0.06±0.04	135.57±4.03	0.13±0.10	0.16±0.13	137.31±4.72	2.96±2.12	2.73±2.03	2.90±1.78	328.11±1.53
	Neural ODE (512)	1.40±0.43	0.65±0.25	424.05±22.54	0.48±0.16	0.68±0.27	332.22±13.02	0.65±0.10	0.53±0.05	1.30±0.05	1770.79±490.00
	MAGI–X (512)	0.12±0.04	0.05±0.02	23.87±0.75	0.13±0.12	0.17±0.17	23.56±0.52	0.37±1.31	0.14±0.13	0.34±0.33	93.19±1.76

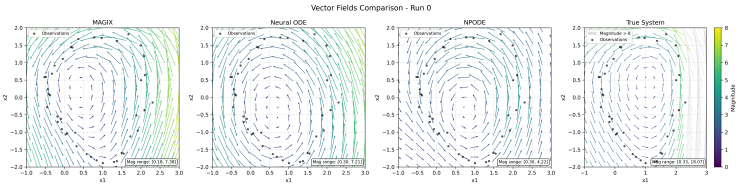
Increasing NPODE to 2000 iterations improves its accuracy and, on LV forecasting, it matches or slightly outperforms MAGI-X (tie on x_1, marginal gain on x_2); however, this comes with a 3–4$\times$ longer runtime (about 135–328 s vs. 24–93 s), whereas MAGI-X remains superior on FN and Hes1. Overall, MAGI-X offers the best accuracy–time trade-off.

D.2 Additional derivative plots

We repeat the derivative analysis for LV (log space) using the same 40×40 grid and error metric $\| \hat{f}(x) - f^\star(x) \|_2^2$ as in Sec. 3.4. Errors are smallest along the observed orbit and remain moderate away from it; the learned vector field reproduces the phase-plane geometry seen in FN (limit cycle and flow orientation).



(a) LV squared error heatmap $\| \hat{f}(x) - f^\star(x) \|_2^2$



(b) LV learned vector field $\hat{f}_{\hat{\theta}}(x)$ (color = $\|\hat{f}_{\hat{\theta}}(x)\|_2$)

Figure 9: Additional LV derivative analyses. Setup matches Sec. 3.4; patterns mirror FN results in the main text.

D.3 Synthetic High Dimensional Systems

Table 5: Means and standard deviations of computational time (seconds) on the synthetic large system example over 100 runs. All experiments are ran under the same CPU setting.

Component Dimension	Computational Time
10	72.26 ± 3.43
20	135.33 ± 2.57
40	272.65 ± 3.77

Table 6: Means and standard deviations of trajetory RMSEs over 100 runs on the synthetic large system example after excluding the divergence cases.

		Reconstructed		Inferred	
Dimension	No. Divergence	Imputation	Forecast	Imputation	Forecast
10	4	0.08 ± 0.05	0.38 ± 0.40	0.06 ± 0.01	0.30 ± 0.25
20	10	0.07 ± 0.02	0.41 ± 0.35	0.06 ± 0.01	0.36 ± 0.28
40	11	0.07 ± 0.01	0.46 ± 0.29	0.07 ± 0.01	0.46 ± 0.30

For the ODEs parameter inference where the derivative function f has known parametric form, Wenk et al. (2020) have demonstrated that the runtime of the *gradient constrained* approach scales linearly in the number of components. We also want to show that such linear scaling computational advantage can be preserved in MAGI-X, in which the parametric form of f is completely *unknown*. For illustration, we apply MAGI-X on a synthetic Hamiltonian system ODEs dynamic with 10, 20, and 40 components. Following the same data generation procedure, we again simulate 41-point noisy observations for each component by adding i.i.d. 0.1^2 variance Gaussian random noise.

Table 5 shows that the experimental runtime of MAGI-X increases linearly in the number of components. The linear scaling of MAGI-X is achieved by (i) its *gradient constrained* framework that circumvents the costly numerical integration and (ii) the use of neural network for approximating the derivative function f. If vector-valued GP is employed for learning f such as in Heinonen et al. (2018) and Ridderbusch et al. (2020), the linear scaling would not be feasible due to the cubic complexity required for computing the inverse of covariance matrix. More surprisingly, the runtime of MAGI-X on the 40-component system is only about 270 seconds (Table 5), while Neural ODE takes more than 300 seconds for training a two-component system (Table 1). Again, this demonstrate the remarkable computational efficiency of MAGI-X on large system inference over the existing state-of-the-art methods that is built upon numerical integration, i.e., the *classical* approach for ODEs dynamic inference.

To study the performance of MAGI-X in the large system setting, we consider the following ODEs derived from the Hamiltonian system $H(q,p) = \frac{1}{2}p^T p + q^T q$,

$$\begin{cases} \dot{p} = -\frac{\partial H}{\partial q} = -2q \\ \dot{q} = \frac{\partial H}{\partial p} = p \end{cases} \tag{42}$$

where q and p are both n-dimensional vector, and thus the system has total of $D = 2n$ components. The initial value of the system is simulated randomly from the standard Gaussian distribution. We consider the ODEs system with $n = 5, 10, 20$, which corresponds to the number of components $D = 10, 20, 40$ for demonstration. Similar to the data generation procedure in the three real world systems, we again inject 0.1^2 variance i.i.d. Gaussian random noise to the groundtruth trajectory to obtain the 41-point noisy observations for each component. MAGI-X is then ran for 5,000 iterations with the default learning rate.

As presented in Table 5 runtime scales linearly in the component dimensions. Given that our optimization could still once in a while stuck at local optimum and yield a divergent forecasting trend as in the LV example (Figure 8), we would expect the same limitation in the larger system example, and thus we include the number of divergence case as another measure for evaluating the performance. We define the divergence to be the case if any component's forecasting trajectory RMSE is greater than 5, which indicates that the forecasting trend is far away from the observation domain of equation 42. Table 6 shows the trajectory RMSEs excluding the divergence cases. We can see that the recovery of the true dynamic is accurate for the imputation phase, but the performance on the forecasting is not as good. Moreover, we do observe more divergence case as the number of components increase, which is expected as the higher dimensional problem is harder. However, even on the 40-component system, we only observe about 10% divergence case, showing that a few restarts would be sufficient to avoid the divergence problem, which is still a significant computational saving from the existing methods (e.g. runtime of MAGI-X on the 40-component system is 270 seconds, while runtime of Neural ODE on the 3-component system is more than 500 seconds).

D.4 UNCERTAINTY QUANTIFICATION VIA DROPOUT AND A NOISE-FREE CALIBRATION CHECK

As a complement to the GP-based uncertainty in the main text, we estimate predictive uncertainty via Monte Carlo (MC) dropout applied to the neural network parameterizing the derivative field f.[1]

Table 7 summarizes empirical coverage and (normalized) band widths across FN, LV, and Hes1. Forecast PIs are well calibrated (90.8–99.4%), whereas fit-region coverage is lower (65.4–76.5%) when training observations are noisy. This under-coverage indicates *overconfidence* in the presence of measurement noise rather than a structural calibration failure.

To isolate the noise effect, we repeat the same procedure with noise-free training trajectories ($\sigma = 0$). Coverage in the fit region improves markedly (FN: 65.4→85.5; LV: 66.0→98.1; Hes1: 76.5→96.3), and forecast coverage remains near-nominal or conservative (FN 99.1, LV 99.7, Hes1 98.3), with bands that are as tight or tighter than in the noisy case; see Table 8. Taken together, these results indicate that the apparent miscalibration under noise is driven by measurement error, not by the MAGI-X UQ mechanism. Overall, MAGI-X yields interpretable predictive uncertainty through both the GP posterior (main text) and MC dropout (this appendix), adapting to system complexity and forecast difficulty.

Table 7: Coverage and normalized width of 90% predictive intervals using MC dropout ($p = 0.1$, $N = 50$).

System	Region	Coverage@90	Width	Norm. Width
FN	Fit	65.4%	0.166	0.054
	Forecast	91.6%	0.301	0.102
LV	Fit	66.0%	0.152	0.048
	Forecast	99.4%	0.529	0.170
Hes1	Fit	76.5%	0.219	0.099
	Forecast	90.8%	0.646	0.302

Table 8: Coverage and normalized width of 90% predictive intervals under noise-free training ($\sigma = 0$).

System	Region	Coverage@90	Width	Norm. Width
FN	Fit	85.5%	0.167	0.054
	Forecast	99.1%	0.278	0.093
LV	Fit	98.1%	0.156	0.050
	Forecast	99.7%	0.457	0.147
Hes1	Fit	96.3%	0.162	0.073
	Forecast	98.3%	0.476	0.219

[1] At test time we retain dropout with rate $p = 0.1$, draw $N = 50$ stochastic rollouts from the learned initial conditions, and form pointwise 90% predictive intervals (PIs) using the empirical 5th/95th percentiles.