Continuous-depth Deep Gaussian Processes

Ying Li[1] **Sanyou Wu**[2] **Zi Yang**[3]
Qiaochu Xu[3] **Yuhao Liu**[4] **Zhidi Lin**[5]
Michael Minyi Zhang[3] **Petar Djuric**[6] **Alexandre H. Thiery**[1*]

[1]National University of Singapore [2]Columbia University
[3]The University of Hong Kong [4]Capital One
[5]The Education University of Hong Kong [6]Stony Brook University

Abstract

Deep Gaussian processes (DGPs) are appealing Bayesian regression models, but their standard discrete-depth parameterization often becomes harder to optimize as depth grows. We argue that depth is better treated as a latent continuous evolution than as a long discrete stack. We therefore propose a continuous-depth formulation in which the latent representation follows a stochastic differential equation with a depth-indexed drift family equipped with a GP-based prior. This reframes the core design choice as the prior placed on that family, rather than adopting a purely neural continuous-depth parameterization as in neural ODE or neural SDE models. We study two instantiations of this idea: CDGP, a direct GP prior over the state–depth drift, and FlowDGP, a flow-evolved prior that captures richer depth dependence at lower training cost. To motivate the redesign, we analyze how the single-sample DGP objective can become increasingly sensitive under repeated layer composition, while residual DGP is a useful first repair that still remains a finite discrete construction. We also give pathwise sensitivity analysis for the continuous-depth formulation. Experiments on synthetic, benchmark, and large-scale regression tasks show that residual DGP improves over plain DGP, while CDGP and FlowDGP provide a stronger overall story across the regimes we study.

1 Introduction

Deep Gaussian processes (DGPs) promise hierarchical probabilistic regression, but in practice they become difficult to optimize as depth grows. This is exactly the regime where a DGP should be most useful, yet the standard discrete stack often becomes least reliable there. The central question is therefore not only how to stabilize a deep composition, but how depth itself should be parameterized in a probabilistic latent-function model.

Empirically, deeper DGPs often exhibit pathological behavior, with additional layers degrading performance [7, 6]. As a result, DGPs are usually kept shallow, which undercuts their original modeling appeal. This practical difficulty is consistent with a simple sensitivity view of discrete composition: repeated layerwise transformations can amplify parameter perturbations and make optimization progressively more brittle.

Prior work clarifies parts of this picture without resolving it. Theoretical studies identified depth pathologies in deep compositions [7, 6], while residual DGP variants [19] introduced the most direct repair by keeping an identity path. Residual DGP is therefore the right place to start, but it still leaves depth represented as a finite stack of separately parameterized layers. Other successful alternatives, such as deep kernel learning [18] or thin DGPs [5], can improve predictive performance, and related continuous-depth constructions such as differential GP flows [10] show that GP-based dynamics can

*Corresponding author: a.h.thiery@nus.edu.sg.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

be useful. Our focus is narrower: preserve a GP-native latent-function view of depth while changing the way depth itself is parameterized.

Plain discrete-depth DGPs admit an optimization pathology: repeated layer composition can make the single-sample training objective increasingly sensitive to parameter perturbations. Residual DGP is the natural first fix because it weakens that amplification through an identity shortcut. But it does not change the underlying parameterization of depth. Once depth is treated as continuous latent evolution, the most natural non-GP alternatives are neural ordinary differential equations (ODE) and neural stochastic differential equations (SDE) models. We study the complementary design in which the deep stack is replaced by a latent continuous-depth system whose drift family retains a GP-based prior. Once this latent SDE is in place, the central design choice is no longer the number of layers, but the prior placed on that drift family. This leads to two concrete models: CDGP, which uses a direct separable GP prior on the drift family, and FlowDGP, which uses a GP prior on a base drift together with flow evolution across depth.

Our contributions are fourfold. First, we propose a continuous-depth methodology for DGP regression in which the latent representation evolves as an SDE with a depth-indexed drift family equipped with a GP-based prior. Second, we study two priors on that drift, CDGP and FlowDGP, within a shared sparse variational inference framework. Third, we identify the repeated Jacobian products that control sensitivity in discrete DGPs and derive a finite-horizon pathwise sensitivity bound for the continuous-depth formulation. The stronger contractive corollary is a specialized auxiliary result, not a property used to design our models. Fourth, we show empirically that residual DGP is a meaningful first repair of plain DGP, and that against both discrete-depth baselines and the natural continuous-depth neural alternatives, CDGP and FlowDGP are more stable or more competitive across toy, benchmark, and scalable regression settings. Continuous depth is a better way to represent depth in deep GP regression than a long discrete stack, especially when the drift family retains GP-based structure rather than being parameterized only by neural dynamics.

2 Background

We review only the ingredients needed for the continuous-depth reformulation: GPs as priors over latent functions, DGPs as discrete compositions of those functions.

Gaussian Process (GP). A GP [15] defines a prior over scalar-valued functions $f : \mathbb{R}^D \to \mathbb{R}$, written $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are the mean and covariance functions. Unless otherwise stated, we assume zero mean and write $k(\mathbf{x}, \mathbf{x}')$ simply as k when no ambiguity arises. For vector-valued mappings $f : \mathbb{R}^D \to \mathbb{R}^P$, we model each output dimension independently: $f_p(\cdot) \sim \mathcal{GP}(0, k)$ for $p = 1, \ldots, P$. Thus f is represented by P scalar-valued GPs sharing the same kernel.

Deep Gaussian Process (DGP). A DGP [4] defines a hierarchy of latent representations through a composition of GP mappings. Let $\mathbf{h}^{(\ell)} \in \mathbb{R}^{D_\ell}$ denote the latent representation at layer ℓ, for $\ell = 1, \ldots, L$, with $\mathbf{h}^{(0)} = \mathbf{x} \in \mathbb{R}^{D_x}$. The observed output $\mathbf{y} \in \mathbb{R}^{D_y}$ is modeled as a noisy observation of the deepest latent representation, $\mathbf{y} = \mathbf{h}^{(L)} + \boldsymbol{\epsilon}_y$ with $\boldsymbol{\epsilon}_y \sim \mathcal{N}(\mathbf{0}, \sigma_y^2 \mathbf{I})$. The latent representations are defined recursively by $\mathbf{h}^{(\ell)} = f^{(\ell)}(\mathbf{h}^{(\ell-1)}) + \boldsymbol{\epsilon}^{(\ell)}$ for $\ell = 1, \ldots, L$, where $\boldsymbol{\epsilon}^{(\ell)} \sim \mathcal{N}(\mathbf{0}, \sigma_\ell^2 \mathbf{I})$ is the additive Gaussian noise at layer ℓ, and $f^{(\ell)} : \mathbb{R}^{D_{\ell-1}} \to \mathbb{R}^{D_\ell}$ is a vector-valued GP mapping. Although this hierarchy increases expressive power, training becomes significantly harder as depth increases [7, 6].

3 From Discrete-Depth Pathology to the Residual Bridge

Depth and Lipschitz Intuition. Repeated composition can amplify parameter perturbations. To characterize this possibility, we analyze the Lipschitz constant of the standard DGP evidence lower bound (ELBO) under variational inference [16]. Since this objective is typically optimized using Monte Carlo estimates, we work with a single-sample estimator:

$$\hat{\mathcal{L}}_{\mathrm{DGP}}(\boldsymbol{\theta}) = \log p(\mathbf{y} \mid \mathbf{f}^{(L)}) - \sum_{\ell=1}^{L} \mathrm{KL}\big[\cdot \,\|\, \cdot\big]. \tag{1}$$

Here, the first term is the data-fit term and the second is a Kullback–Leibler (KL) regularizer. We suppress the KL arguments because their exact form depends on the chosen approximation. More-

over, $\boldsymbol{\theta}$ denotes all variational parameters and kernel hyperparameters, and $\mathbf{f}^{(L)2}$ is sampled from the variational distribution over the final-layer latent function values, given by

$$\mathbf{f}^{(L)} = f^{(L)}\big(f^{(L-1)}(\cdots f^{(1)}(\mathbf{x})\cdots)\big).$$

Let L_{DGP} denote the Lipschitz constant of $\hat{\mathcal{L}}_{\mathrm{DGP}}$ on the parameter domain under consideration, with Monte Carlo randomness held fixed. The following proposition gives an assumption-dependent upper bound on this sensitivity.

Proposition 3.1. *Consider an L-layer DGP whose sampled layer map at layer ℓ is λ_ℓ-Lipschitz with respect to its input. Assume in addition that the output-gradient term and the layerwise parameter-Jacobian term are uniformly bounded, and that the KL regularizer decomposes layerwise with uniformly bounded block gradients, as stated precisely in Appendix A. Then the Lipschitz constant of the single-sample DGP objective satisfies*

$$L_{\mathrm{DGP}} \le C\left(\sum_{k=1}^{L}\prod_{\ell=k+1}^{L}\lambda_\ell^2\right)^{1/2} + O(\sqrt{L}),$$

where C is independent of L, and the $O(\sqrt{L})$ term captures the contribution of the KL regularizer under that layerwise decomposition. The proof is given in Appendix A.

Proposition 3.1 shows that the depth dependence of L_{DGP} is driven mainly by the data-fit term, rather than the KL regularization, and may grow exponentially with depth. Even at the level of a single-sample pathwise estimator, this implies increasing sensitivity to parameter perturbations as depth grows, making standard DGP optimization progressively more ill-conditioned, with estimator variance potentially worsening the effect. Repeated discrete composition allows small perturbations to be multiplied from one layer to the next. That is the pathology we want to repair before introducing a continuous-depth model.

Residual DGP as a First Repair. This motivates a residual formulation of DGPs [19], in which each layer updates the current latent representation additively rather than fully remapping it:

$$\mathbf{h}^{(\ell)} = \mathbf{h}^{(\ell-1)} + f^{(\ell)}(\mathbf{h}^{(\ell-1)}) + \boldsymbol{\epsilon}^{(\ell)}, \qquad \ell = 1, \ldots, L.$$

Because the identity path is retained, each residual function only needs to model a correction, reducing perturbation amplification at each layer. However, the model remains a finite discrete composition, so its sensitivity can still be influenced by repeated Jacobian products and may accumulate with depth. Residual DGP is therefore the right first repair, but it still treats depth as a finite stack of separately parameterized residual blocks. This motivates the continuous-depth redesign introduced in Section 4.

4 Continuous-Depth Modeling and Inference

If depth is meant to express gradual representation evolution, then a finite collection of separately parameterized residual layers is only one way to represent that evolution. We therefore replace the discrete layer index by a depth variable $t \in [0, T]$, so the latent representation evolves gradually rather than through a fixed stack of residual blocks.

The resulting model is a redesign motivated by the residual view, not a claimed scaling limit of a discrete DGP. Its advantage is that it separates the latent dynamics from the choice of parameterization across depth. Once this shared continuous-depth system is fixed, the main modeling question becomes what prior should be placed on its depth-indexed drift family.

Concretely, the latent representation evolves over depth $t \in [0, T]$ according to the Itô SDE

$$d\mathbf{h}_t = f_t(\mathbf{h}_t)\, dt + \sigma_t\, d\mathbf{w}_t, \tag{2}$$

where $\mathbf{w}_t$ is a Wiener process and $\sigma_t\, d\mathbf{w}_t$ is the continuous counterpart of the layerwise noise $\boldsymbol{\epsilon}^{(\ell)}$.[3] The central modeling object is therefore the depth-indexed drift family $\{f_t\}_{t\in[0,T]}$, with each f_t

[2]Throughout the paper, we use the same notation for Monte Carlo samples and their underlying random variables or function values whenever the distinction is clear from context.

[3]If discretized by Euler–Maruyama with unit step size, Eq. (2) has the update form of a residual DGP layer, but uses a shared depth-indexed drift prior rather than independent layer GPs.

acting as a vector field on the current latent state. This formulation cleanly separates the latent dynamics from the way dependence across depth is parameterized.

The paper studies two GP-based priors on this same drift family. CDGP places a direct GP prior on the joint state-depth drift, while FlowDGP generates the family by placing a GP prior on a base drift and evolving it across depth. The common latent SDE is the same in both cases; only the prior on $\{f_t\}$ changes.

4.1 CDGP: Separable Kernel Instantiation

The continuous-depth model in Section 4 leaves open the prior on the drift family $\{f_t\}$. The most direct answer is to write that family as a single function of state and depth, $f_t(\mathbf{h}) \equiv f(\mathbf{h}, t)$, and place a GP prior on this joint state–depth drift with a separable kernel:

$$k\big((\mathbf{h}, t), (\mathbf{h}', t')\big) = k_h(\mathbf{h}, \mathbf{h}')\, k_t(t, t').$$

We term this first prior the continuous-depth DGP (CDGP). CDGP is the most direct implementation of the continuous-depth idea: place a GP prior on the drift family itself and infer that object with standard sparse-GP machinery. The separable structure enables a Kronecker factorization of the inducing-point covariance, making inference scalable. However, it restricts how the drift can vary jointly with state and depth: the covariance between drift values at different depths factorizes into a product of state-only and depth-only terms, preventing the kernel from capturing interactions in which the spatial structure of the drift changes with depth. This motivates a second, more structured prior on the same family $\{f_t\}$.

4.2 FlowDGP: CNF-Transformed GP

CDGP models the shared drift family directly, but its separable kernel is deliberately restrictive. We therefore consider a second prior on the same family $\{f_t\}$, one that enriches the depth dependence without incurring the cost of a non-separable joint kernel. We draw a base drift f_0 from a GP prior and evolve it across depth via a continuous normalizing flow (CNF). A shared vector field $g_{\boldsymbol{\phi}}$ defines an ODE whose solution at depth t gives the evolved drift f_t. Here $g_{\boldsymbol{\phi}}$ acts on a finite-dimensional representation of the drift itself, not on the latent state $\mathbf{h}_t$; in practice it is applied pointwise to sampled drift values such as $f_0(\mathbf{x}_n)$, and we parameterize it as a small multilayer perceptron (MLP) with parameters $\boldsymbol{\phi}$. For regression with latent state dimension D_y, each such pointwise drift value lies in $\mathbb{R}^{D_y}$, so $g_{\boldsymbol{\phi}} : \mathbb{R}^{D_y} \to \mathbb{R}^{D_y}$. Concretely,

$$\frac{\mathrm{d}f_t}{\mathrm{d}t} = g_{\boldsymbol{\phi}}(f_t), \qquad f_0 \sim \mathcal{GP}(0, k_0). \tag{3}$$

Here f_0 is learned through its inducing variables rather than fixed as a function. The displayed equation is a function-level idealization; in inference we apply the same flow pointwise to sampled drift values along each propagated data trajectory.

FlowDGP should therefore be understood as a second prior on the same latent SDE in Eq. (2), not as a separate model family. The contrast is simple: CDGP parameterizes depth dependence directly in the kernel, whereas FlowDGP parameterizes it by evolving a base drift through the neural flow $g_{\boldsymbol{\phi}}$. The evolution is deterministic given f_0, keeping inference on a single base GP. This avoids both the large joint kernel matrix of a non-separable kernel and the additional path inference required by a diffusion over f_t. The choice is motivated by tractability, rather than optimality among flow or diffusion architectures. Depth dependence is instead captured by the ODE integration of $g_{\boldsymbol{\phi}}$, which adds solver cost but, as we show in Section 5.2, yields substantially faster training than CDGP. This construction is also related in spirit to differential GP flows [10], but here it is used to define a prior over the depth-indexed drift that sits inside the same continuous-depth regression framework as CDGP.

4.3 Variational Inference

CDGP and FlowDGP are two priors on the same continuous-depth latent system. Their shared inference structure is most transparent at the level of the common simulator rather than model-specific algebra. For each input $\mathbf{x}_n$, both models produce a terminal latent state $\mathbf{h}_T^{(n)}$ and then apply the same readout likelihood:

$$\mathbf{y}_n \sim p(\mathbf{y}_n \mid \mathbf{h}_T^{(n)}), \qquad \mathbf{h}_T^{(n)} = \Phi(\mathbf{x}_n, \mathbf{F}, \xi_n; \boldsymbol{\theta}), \tag{4}$$

where $\mathbf{F}$ denotes the latent GP object, ξ_n collects the Gaussian exogenous noise used during path simulation, and $\boldsymbol{\theta}$ collects the remaining model, kernel, and variational parameters. Here $\Phi(\mathbf{x}_n, \mathbf{F}, \xi_n; \boldsymbol{\theta})$ denotes the deterministic map that propagates the latent state from depth 0 to depth T once $\mathbf{F}$ and ξ_n are given.

The two models differ only in what object carries the GP prior. In CDGP, the latent GP object is the joint state–depth drift $\mathbf{F} = f(\mathbf{h}, t)$, so inference is performed directly on the drift defined over the joint state–depth domain. In FlowDGP, the latent GP object is the base drift $\mathbf{F} = f_0$; the full depth-indexed family (f_t) is then obtained deterministically by evolving f_0 through the flow ODE in Eq. (3). This is why FlowDGP places inducing points only on the base GP, whereas CDGP places them on the joint state–depth input.

Both models use the same variational recipe. Introduce inducing variables $\mathbf{U}$ for the latent GP object $\mathbf{F}$, place a Gaussian variational distribution

$$q(\mathbf{U}) = \mathcal{N}(\mathbf{m}, \mathbf{S}), \tag{5}$$

and keep the GP conditional exact, so that the full approximation takes the form

$$q(\mathbf{F}, \mathbf{U}, \xi) = q(\mathbf{U})\, p(\mathbf{F} \mid \mathbf{U})\, p(\xi). \tag{6}$$

All model-specific structure remains inside the simulator Φ: once $\mathbf{F}$ is sampled from its sparse variational posterior and ξ_n is sampled from its Gaussian law, the latent path and the terminal state are obtained by deterministic propagation. The evidence lower bound is

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{U})\, p(\mathbf{F}|\mathbf{U})\, p(\xi_n)} \Big[\log p\big(\mathbf{y}_n \mid \Phi(\mathbf{x}_n, \mathbf{F}, \xi_n; \boldsymbol{\theta})\big) \Big] - \mathrm{KL}\big[q(\mathbf{U}) \,\|\, p(\mathbf{U})\big]. \tag{7}$$

This is the common inference structure: inducing-point variational inference for GP-driven continuous-depth simulators. In practice, Φ is implemented on the discretized depth grid $0 = \tau_0 < \tau_1 < \cdots < \tau_J = T$ using a left-point Euler–Maruyama scheme. Writing $\Delta\tau_j = \tau_j - \tau_{j-1}$ and $\Delta\mathbf{w}_j = \mathbf{w}_{\tau_j} - \mathbf{w}_{\tau_{j-1}}$, one simulated path is generated by first sampling the relevant GP object and then updating

$$\mathbf{h}_{\tau_j}^{(n)} = \mathbf{h}_{\tau_{j-1}}^{(n)} + f_{\tau_{j-1}}^{(n)}\big(\mathbf{h}_{\tau_{j-1}}^{(n)}\big)\, \Delta\tau_j + \sigma_{\tau_j} \Delta\mathbf{w}_j, \qquad j = 1, \ldots, J, \tag{8}$$

starting from $\mathbf{h}_0^{(n)} = \mathbf{x}_n$. In CDGP, the drift value $f_{\tau_{j-1}}^{(n)}(\mathbf{h}_{\tau_{j-1}}^{(n)})$ is read directly from the sampled joint GP $f(\mathbf{h}, t)$; in FlowDGP, one first samples the base GP f_0, transports it through the flow ODE to obtain (f_t), and then uses the transported drift in the same update.

With S Monte Carlo samples, the optimized discretized objective is

$$\widehat{\mathcal{L}} = \sum_{n=1}^{N} \frac{1}{S} \sum_{s=1}^{S} \log p\big(\mathbf{y}_n \mid \mathbf{h}_T^{(n,s)}\big) - \mathrm{KL}\big[q(\mathbf{U}) \,\|\, p(\mathbf{U})\big], \tag{9}$$

where each $\mathbf{h}_T^{(n,s)}$ is obtained by the corresponding simulated path under Eq. (8).

For CDGP, we instantiate this construction with $\mathbf{U} = f(\widetilde{\mathbf{Z}})$ on the Cartesian product $\widetilde{\mathbf{Z}} = \mathbf{Z}_h \times \mathbf{Z}_t$, where the separable kernel gives the Kronecker prior $\mathbf{K}_{uu} = \mathbf{K}_h(\mathbf{Z}_h, \mathbf{Z}_h) \otimes \mathbf{K}_t(\mathbf{Z}_t, \mathbf{Z}_t)$. The simulator Φ propagates the latent state by discretizing the SDE in Eq. (2) and evaluating the sampled drift $f(\mathbf{h}_{\tau_{j-1}}^{(n)}, \tau_{j-1})$ along the path. For FlowDGP, we instantiate the same construction with $\mathbf{u} = f_0(\mathbf{Z})$ for the base GP alone. After sampling the base drift from its sparse variational posterior, Φ first transports it across depth through the ODE in Eq. (3) to obtain the family (f_t), and then propagates the latent state with those transported drift values. CDGP and FlowDGP therefore share the same likelihood, the same simulator-level variational logic, and the same Monte Carlo training principle; they differ only in the latent GP object being approximated and in the domain on which inducing variables are placed.

4.4 Optimization Stability

We now ask what kind of sensitivity statement survives once the deep stack is replaced by a latent SDE. The relevant quantity is the pathwise sensitivity of the SDE solution to parameter perturbations. This analysis clarifies what replaces the repeated Jacobian products of a discrete stack, but it is not itself a training guarantee for CDGP or FlowDGP.

Theorem 4.1. *Consider the SDE $d\mathbf{h}_t = f(\mathbf{h}_t, t; \boldsymbol{\theta})\, dt + \sigma_t\, d\mathbf{w}_t$. Let $\mathbf{h}_t^{(1)}$ and $\mathbf{h}_t^{(2)}$ denote the strong solutions associated with parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively, driven by the same Brownian motion. Assume that the diffusion coefficient σ_t is shared across the two systems, and that f is Lipschitz continuous in the state variable with constant L_h, and in the parameter variable with constant L_θ. Then, for any $t \in [0, T]$,*

$$\left\|\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(2)}\right\| \leq L_\theta\, \phi(L_h, t) \left\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\right\|, \tag{10}$$

where $\phi(L_h, t) = (e^{L_h t} - 1)/L_h$ for $L_h > 0$, with continuous extension $\phi(0, t) = t$, for almost every sample path. See Appendix B for the proof.

Theorem 4.1 is a generic pathwise sensitivity statement. Unlike the discrete-depth proposition, it does not grow through repeated products of layerwise Jacobians. It still carries an exponential dependence on T, however, so a depth-uniform conclusion requires extra structure.

Corollary 4.2. *Under the assumptions of Theorem 4.1, suppose in addition that the drift satisfies a one-sided contractivity condition on $[0, T]$, i.e., there exists $\mu > 0$ such that, for all $\mathbf{h}_1, \mathbf{h}_2$ and $t \in [0, T]$,*

$$\left\langle f(\mathbf{h}_1, t; \boldsymbol{\theta}) - f(\mathbf{h}_2, t; \boldsymbol{\theta}), \mathbf{h}_1 - \mathbf{h}_2 \right\rangle \leq -\mu \|\mathbf{h}_1 - \mathbf{h}_2\|^2. \tag{11}$$

Then,

$$\|\mathbf{h}_T^{(1)} - \mathbf{h}_T^{(2)}\| \leq \frac{L_\theta}{\mu} \sqrt{1 - e^{-\mu T}}\, \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| \leq \frac{L_\theta}{\mu} \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|, \tag{12}$$

for almost every sample path. See Appendix C for the proof.

Under the contractive regime, the pathwise sensitivity is bounded by L_θ/μ. If the loss gradient is bounded by C, the data-fit term at terminal time satisfies, denoting by L_{SDE} the Lipschitz constant of the terminal data-fit term with respect to $\boldsymbol{\theta}$,

$$L_{\mathrm{SDE}} \leq \frac{C\, L_\theta}{\mu}. \tag{13}$$

The qualitative difference from discrete DGPs is the object that controls sensitivity. Proposition 3.1 is driven by repeated discrete composition, whereas the continuous-depth formulation yields bounds in terms of a single drift field. The contractive corollary is intentionally stronger and more specialized: neither CDGP nor FlowDGP explicitly enforces that regime. We therefore use these results to interpret the redesign, while Section 5.1 shows empirically that the proposed continuous-depth models remain stable across the depth ranges we study. Appendix F.2 provides a controlled SDE illustration of Corollary 4.2.

5 Experiments

The experiments are organized as a sequence of evidence for the paper's main claim. First, the toy study makes the discrete-depth pathology visible and tests whether residualization is enough to repair it. Second, the benchmark study asks whether the continuous-depth redesign remains competitive on realistic regression tasks once residual DGP is granted as the strongest discrete-depth baseline. Third, the large-scale study asks which part of the continuous-depth family remains practical when direct GP baselines become too expensive to train.

5.1 Toy Example

We begin with a synthetic regression task designed to expose the depth pathology directly. The point of this experiment is diagnostic: make the failure of plain DGP visible, show that residualization is the right first repair, and then ask whether the continuous-depth models remain stable beyond it. The dataset comprises 800 input–output pairs generated from a nonlinear function with additive Gaussian noise, partitioned into 300 training and 500 test samples. We report test negative log-likelihood (NLL), root mean squared error (RMSE), and mean absolute error (MAE), averaged over random seeds. Details of the dataset and hyperparameter settings are provided in Appendix E.1 and Appendix E.2, respectively.

Figure 1 should be read row by row. The top row tests the failure mode of increasing discrete depth, the middle row asks whether the continuous-depth models stay stable as the numerical resolution J changes, and the bottom row asks whether they remain well behaved as the terminal depth T increases.

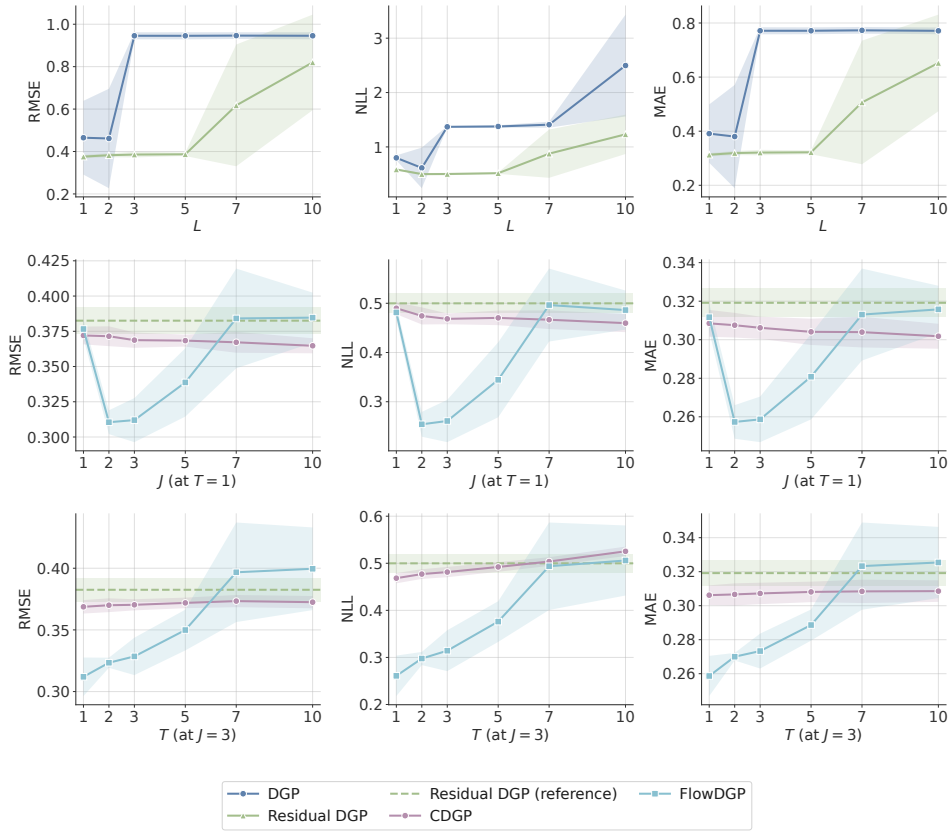


Figure 1: **Toy regression results.** Top: test root mean squared error (RMSE), negative log-likelihood (NLL), and mean absolute error (MAE) versus depth L for standard DGP and residual DGP. Middle: CDGP and FlowDGP versus J at fixed $T = 1$, with the dashed line showing the two-layer residual DGP reference. Bottom: CDGP and FlowDGP versus T at fixed $J = 3$. Shaded regions denote one standard deviation across random seeds.

The top row of Figure 1 shows DGP and Residual DGP as the number of layers L increases from 1 to 10. Standard DGP deteriorates sharply across all three metrics at larger L, consistent with the exponential Lipschitz growth identified in Proposition 3.1. Residual DGP is more robust due to its identity shortcut but still degrades at larger depths, confirming that residualization helps materially but does not remove the limitations of discrete composition.

The middle row fixes $T = 1$ and varies the number of discretization steps J from 1 to 10. CDGP remains stable across all metrics, matching or outperforming the two-layer Residual DGP configuration (dashed line). FlowDGP achieves lower NLL than CDGP at $J \leq 5$.

The bottom row fixes $J = 3$ and varies T from 1 to 10. CDGP maintains stable performance as T increases, and FlowDGP shows better RMSE and MAE at T values. This behavior is qualitatively consistent with the sensitivity perspective of Section 4.4, but the empirical point stands on its own: once depth is parameterized continuously, the models remain well behaved across the range we test.

5.2 Benchmark Datasets

The benchmark study asks a different question from the toy experiment: once residual DGP is granted as the strongest discrete-depth repair, are the proposed continuous-depth models still competitive on realistic regression tasks? We therefore evaluate all methods on six regression datasets spanning diverse data characteristics: `heavy_tailed`, `heteroscedastic`, and `nonstationary` are synthetic datasets designed to test specific modeling challenges, while `sarcos_small`, `uci_bike_small`, and `uci_energy_small` are subsampled versions of real-world datasets (see Appendix E.1 for details). We report test NLL, RMSE, and training time averaged over random seeds. The benchmark baselines are sparse GP, heteroscedastic GP [11], standard DGP, residual DGP [19], deep kernel learning (DKL) [18], thin DGP (TDGP) [5], neural ODE (NODE) [3], neural SDE (NSDE) [12], and transformed GP (TGP) [13].

Table 1: Regression results on benchmark datasets. We report mean ± std over multiple random seeds. Bold indicates the best result; underline indicates the second best. Training time is in seconds. Rows highlighted in light blue correspond to our methods. Method abbreviations: DKL = deep kernel learning, TDGP = thin DGP, TGP = transformed GP, NODE = neural ODE, and NSDE = neural SDE.

		Dataset					
Metric	**Method**	`heavy_tailed`	`heteroscedastic`	`nonstationary`	`sarcos_small`	`uci_bike_small`	`uci_energy_small`
RMSE ↓	FlowDGP	$\underline{0.235 \pm 0.003}$	$\mathbf{0.819 \pm 0.011}$	0.377 ± 0.123	$\underline{0.323 \pm 0.028}$	$\mathbf{0.934 \pm 0.121}$	0.804 ± 0.009
	CDGP	$\mathbf{0.231 \pm 0.005}$	0.856 ± 0.077	0.634 ± 0.008	0.332 ± 0.004	1.122 ± 0.012	$\underline{0.802 \pm 0.001}$
	Heteroskedastic GP	0.306 ± 0.002	$\underline{0.823 \pm 0.006}$	0.654 ± 0.008	0.345 ± 0.010	1.152 ± 0.006	$\mathbf{0.797 \pm 0.001}$
	Residual DGP	0.311 ± 0.005	$\underline{0.823 \pm 0.003}$	0.676 ± 0.056	0.347 ± 0.006	1.126 ± 0.005	$\underline{0.802 \pm 0.001}$
	Sparse GP	0.309 ± 0.001	1.023 ± 0.008	0.756 ± 0.006	1.051 ± 0.003	1.162 ± 0.004	$\underline{0.802 \pm 0.002}$
	TGP	0.305 ± 0.003	0.893 ± 0.006	0.464 ± 0.008	0.383 ± 0.028	0.986 ± 0.026	0.827 ± 0.004
	DGP	0.321 ± 0.002	1.012 ± 0.002	0.745 ± 0.002	1.050 ± 0.001	1.182 ± 0.031	$\underline{0.802 \pm 0.027}$
	NODE	0.544 ± 0.197	0.897 ± 0.036	0.356 ± 0.029	0.351 ± 0.010	0.993 ± 0.075	0.964 ± 0.099
	TDGP	0.307 ± 0.008	0.894 ± 0.078	$\underline{0.271 \pm 0.091}$	0.393 ± 0.014	1.037 ± 0.034	$\underline{0.802 \pm 0.001}$
	DKL	0.437 ± 0.029	0.928 ± 0.018	0.548 ± 0.065	0.543 ± 0.162	$\underline{0.956 \pm 0.175}$	1.593 ± 0.161
	NSDE	0.480 ± 0.043	0.962 ± 0.037	$\mathbf{0.211 \pm 0.010}$	$\mathbf{0.302 \pm 0.007}$	1.052 ± 0.033	1.317 ± 0.146
NLL ↓	FlowDGP	$\underline{0.329 \pm 0.022}$	$\mathbf{1.248 \pm 0.008}$	$\mathbf{0.411 \pm 0.286}$	1.352 ± 0.692	1.868 ± 0.203	1.229 ± 0.021
	CDGP	0.355 ± 0.037	1.285 ± 0.076	0.979 ± 0.006	$\mathbf{0.376 \pm 0.011}$	1.585 ± 0.016	1.223 ± 0.003
	Heteroskedastic GP	0.490 ± 0.015	1.287 ± 0.005	1.013 ± 0.012	$\underline{0.440 \pm 0.025}$	$\underline{1.569 \pm 0.006}$	1.258 ± 0.011
	Residual DGP	0.490 ± 0.022	1.254 ± 0.004	1.032 ± 0.076	$\underline{0.440 \pm 0.017}$	1.605 ± 0.010	1.232 ± 0.004
	Sparse GP	0.501 ± 0.005	$\underline{1.253 \pm 0.006}$	0.987 ± 0.008	1.470 ± 0.001	1.605 ± 0.002	$\underline{1.211 \pm 0.001}$
	TGP	$\mathbf{0.258 \pm 0.007}$	1.326 ± 0.008	$\underline{0.897 \pm 0.005}$	0.707 ± 0.018	$\mathbf{1.519 \pm 0.053}$	1.257 ± 0.007
	DGP	0.536 ± 0.036	1.442 ± 0.002	1.125 ± 0.002	1.470 ± 0.001	1.800 ± 0.022	1.218 ± 0.003
	NODE	0.870 ± 0.427	42.231 ± 9.592	30.059 ± 3.559	26.245 ± 3.366	52.447 ± 23.808	21.634 ± 11.721
	TDGP	0.357 ± 0.051	5.317 ± 1.026	1.080 ± 1.323	1.278 ± 0.103	5.585 ± 0.573	$\mathbf{1.204 \pm 0.002}$
	DKL	25.536 ± 2.319	34.401 ± 3.518	6.608 ± 0.271	6.605 ± 0.361	4.604 ± 1.652	9.590 ± 1.391
	NSDE	0.828 ± 0.259	3.527 ± 1.601	2.058 ± 0.462	9.516 ± 0.390	1.584 ± 0.049	$\mathbf{1.204 \pm 0.003}$
Train Time	FlowDGP	27.490 ± 0.940	19.130 ± 2.240	27.690 ± 10.150	14.870 ± 1.694	12.887 ± 0.231	31.910 ± 7.909
	CDGP	201.330 ± 1.130	189.180 ± 4.420	126.930 ± 1.670	202.643 ± 12.019	93.149 ± 9.607	178.598 ± 23.021
	Heteroskedastic GP	21.870 ± 0.600	26.060 ± 1.910	22.600 ± 0.730	23.375 ± 5.970	26.174 ± 8.227	31.023 ± 7.772
	Residual DGP	47.880 ± 4.790	74.580 ± 8.340	67.960 ± 2.340	48.065 ± 2.814	51.406 ± 6.523	117.847 ± 16.674
	Sparse GP	11.810 ± 0.160	11.300 ± 1.280	11.510 ± 0.270	10.052 ± 1.869	9.191 ± 0.155	9.236 ± 0.264
	TGP	129.878 ± 8.237	129.303 ± 7.154	137.329 ± 5.959	134.859 ± 8.759	140.042 ± 9.254	130.997 ± 9.929
	DGP	71.360 ± 4.350	124.420 ± 6.030	119.190 ± 2.410	70.540 ± 5.532	71.327 ± 8.291	128.722 ± 11.922
	NODE	13.470 ± 0.230	10.590 ± 0.480	3.840 ± 0.050	4.822 ± 0.283	6.927 ± 2.081	5.573 ± 2.423
	TDGP	52.386 ± 6.236	37.278 ± 15.171	34.389 ± 24.178	32.167 ± 21.912	26.451 ± 2.199	72.789 ± 12.371
	DKL	6.570 ± 0.460	6.880 ± 1.000	8.990 ± 0.540	8.854 ± 0.456	4.051 ± 0.223	3.995 ± 0.044
	NSDE	74.350 ± 9.580	67.580 ± 9.380	69.420 ± 6.630	73.253 ± 2.473	63.443 ± 8.320	79.741 ± 10.047

Table 1 summarizes the results. Three patterns matter most. First, the continuous-depth models are not only stable in the toy setting; they are genuinely competitive on realistic regression tasks. FlowDGP achieves the best NLL on `heteroscedastic` and `nonstationary`, while CDGP achieves the best NLL on `sarcos_small`. TGP has the best NLL on `heavy_tailed` and `uci_bike_small`; CDGP or FlowDGP has lower RMSE than TGP on all six datasets. Second, the comparison to discrete-depth baselines matches the motivating story of the paper: standard DGP performs poorly across most datasets, especially in NLL, and residual DGP improves over DGP but still trails CDGP and FlowDGP. Third, the two continuous-depth variants occupy different practical regimes. FlowDGP is substantially faster than CDGP, with roughly 5–$14\times$ lower training time across datasets, while CDGP remains the strongest direct GP instantiation of the framework.

The benchmark comparison also includes NODE and NSDE, which share continuous-depth dynamics but parameterize the vector field with a neural network rather than a GP-based prior. They play different roles. NODE is the clean deterministic baseline for continuous depth without a GP-based prior, but its NLL is often dramatically worse, especially on `heteroscedastic`, `nonstationary`, and `sarcos_small`. NSDE is the closer stochastic rival: it achieves very strong RMSE on `nonstationary` and `sarcos_small`, but its NLL remains markedly weaker. The benchmark results show more than the usefulness of continuous dynamics alone. In this small- to medium-scale regime, the GP-based prior over the drift family materially improves probabilistic performance relative to the neural continuous-depth alternatives. Together with the toy study, this closes the main comparison against both residual DGP and the non-GP continuous-depth baselines. The next subsection isolates the larger-scale regime in which FlowDGP, rather than CDGP, becomes the practical representative of the continuous-depth approach.

5.3 Large-Scale Datasets

The large-scale study isolates the practical regime of FlowDGP. We compare against methods with neural-network components that scale naturally to larger datasets: DKL [18], neural ODE [3], neu-

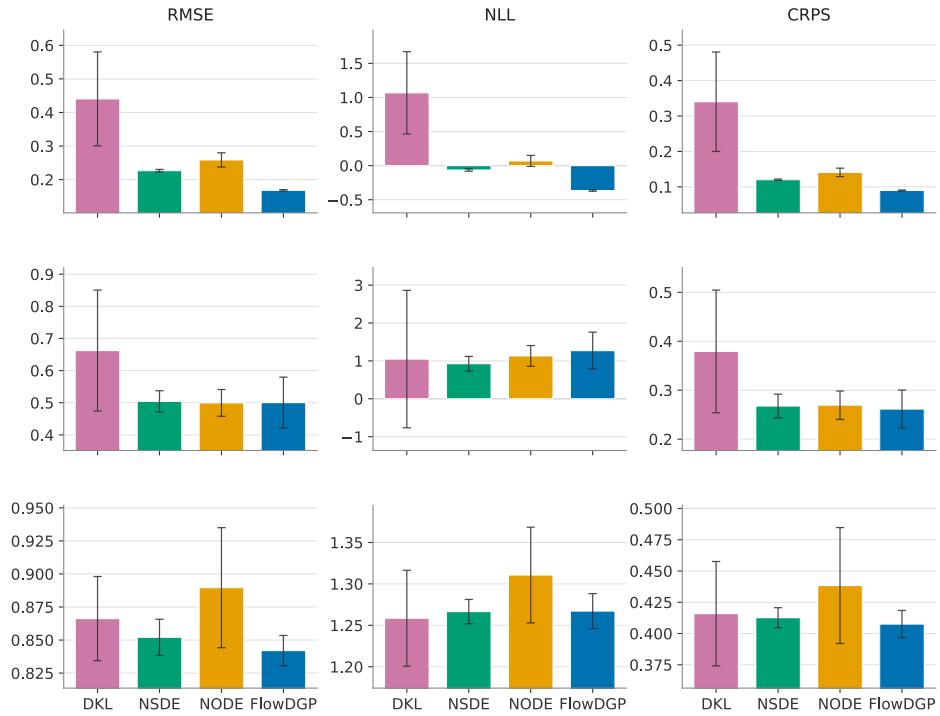


Figure 2: Results on full-size `uci_bike` (top), `uci_energy` (middle), and `sarcos` (bottom) datasets comparing FlowDGP, deep kernel learning (DKL), neural ordinary differential equations (NODE), and neural stochastic differential equations (NSDE). We report root mean squared error (RMSE), negative log-likelihood (NLL), and continuous ranked probability score (CRPS) averaged over random seeds; error bars denote one standard deviation. For readability, the NLL values of DKL on `sarcos` are scaled by 0.1 (first) and $1/3$ (second) to avoid distorting the y-axis scale.

ral SDE [12], and FlowDGP. The evaluation uses the full-size versions of `uci_bike`, `uci_energy`, and `sarcos` (Appendix E.1). CDGP and residual DGP are omitted because their training cost is prohibitive at these scales; the benchmark timings already show this trend, and Appendix D explains why. We report RMSE, NLL, and continuous ranked probability score (CRPS), averaged over random seeds. The point is not that CDGP disappears from the story, but that the two continuous-depth variants occupy different cost-quality regimes: CDGP is the direct GP instantiation, whereas FlowDGP is the scalable one. In this regime, the most meaningful comparison is between FlowDGP and NSDE: both are stochastic continuous-depth models, but only FlowDGP retains a GP-based prior over the drift family.

Figure 2 presents the results. On `uci_bike`, FlowDGP achieves the best performance across RMSE, NLL, and CRPS. On `uci_energy`, it obtains the best CRPS and is tied for the best RMSE, while NSDE has the best NLL. On `sarcos`, it performs comparably to, or slightly better than, neural SDE across all three metrics, while neural ODE and DKL provide the weakest probabilistic performance.

Overall, the large-scale picture is sharper than a generic baseline comparison. NODE remains a useful deterministic continuous-depth reference, but it is not the main rival in probabilistic regression. NSDE is the closer stochastic comparator, and the figure shows that FlowDGP remains competitive with it in NLL and CRPS across the datasets studied here. The practical point is therefore not just that FlowDGP is competitive in point prediction, but that a GP-based prior over the drift family remains useful even in the scalable continuous-depth regime.[4]

6 Related Work

DGPs compose GP mappings [4], with scalable variational inference [16]. Residual [19] and thin DGPs [5] modify this hierarchy. Deep kernel learning [18] instead learns neural features, while

[4] Additional results on YearPredictionMSD are provided in Appendix F.1.

TGPs [13] transform a GP through an invertible, potentially input-dependent map. Our focus is the parameterization of latent depth.

GP networks [9] put GP priors on conditional relations in a discrete graphical model; nonlinear multifidelity GPs [14] learn relations between observed fidelity levels. Our depth variable is latent and continuous, without observations required at intermediate depths.

Neural ODEs and SDEs [3, 12] parameterize continuous-depth dynamics with neural vector fields. Deep state-space GPs [20] use conditional SDE representations over the input time variable within a finite hierarchy, enabling sequential inference. Differential GP flows [10] use a GP posterior to determine the drift and diffusion of an input-warping SDE, followed by a separate output GP. In FlowDGP, a sampled base drift is evolved across latent depth, the state diffusion is specified separately, and the terminal state enters the likelihood directly.

7 Conclusion

For probabilistic regression, depth in a deep GP is better treated as latent continuous evolution than as a long discrete stack. Plain DGPs become increasingly awkward as depth grows, and residual DGP is the right first repair but not the final answer.

We therefore replaced the deep stack by a latent SDE with a depth-indexed drift family and studied two GP-based priors on that family: CDGP as a direct separable prior, and FlowDGP as a structured flow-evolved prior, both trained with a shared sparse variational framework. The theory explains why this continuous-depth formulation lives in a different sensitivity regime from repeated discrete composition, while the strongest depth-uniform statement remains conditional.

Empirically, the pattern is consistent across scales. Residual DGP improves over plain DGP, but the continuous-depth models are more stable or more competitive in the regimes we study, with FlowDGP performing best at larger scale. Continuous depth is a useful redesign of how depth is represented in probabilistic regression. Relative to the natural continuous-depth neural alternatives, especially neural ODEs and neural SDEs, the advantage is not continuous depth alone but continuous depth with a GP-based prior over the drift family.

8 Limitations

On the modelling side, CDGP's separable kernel limits the joint state–depth dependence of the drift, while FlowDGP captures this dependence through a deterministic CNF, so uncertainty in the drift across depths is inherited solely from the base GP f_0. Our empirical evaluation also focuses on regression tasks of moderate dimensionality, and extending the approach to classification or structured prediction is left to future work.

Acknowledgments and Disclosure of Funding

Ying Li and Alexandre Thiery acknowledge support from Singapore MOE AcRF Tier 2 grants MOE-T2EP20123-0010 and MOE-T2EP20223-0011. The contribution of Petar Djuric is supported by the National Science Foundation (NSF) under Award 2212506. The contribution of Michael Zhang is funded by the Research Grants Council under the General Research Fund #17312826 and by the University of Hong Kong Seed Fund for PI Research – Basic Research #2402101367.

References

[1] Thierry Bertin-Mahieux. Year Prediction MSD. UCI Machine Learning Repository, 2011. DOI: 10.24432/C50K61.

[2] Luis Candanedo. Appliances Energy Prediction. UCI Machine Learning Repository, 2017. DOI: 10.24432/C5VC8G.

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[4] Andreas Damianou and Neil D. Lawrence. Deep Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 207–215, 2013.

[5] Daniel Augusto De Souza, Alexander Nikitin, ST John, Magnus Ross, Mauricio A Álvarez, Marc Deisenroth, João Paulo Gomes, Diego Mesquita, and César Lincoln Mattos. Thin and deep Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[6] Matthew M Dunlop, Mark A Girolami, Andrew M Stuart, and Aretha L Teckentrup. How deep are deep Gaussian processes? *Journal of Machine Learning Research*, 19(54):1–46, 2018.

[7] David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 202–210, 2014.

[8] Hadi Fanaee-T. Bike sharing. UCI Machine Learning Repository, 2013. DOI: 10.24432/C5W894.

[9] Nir Friedman and Iftach Nachman. Gaussian process networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[10] Pashupati Hegde, Markus Heinonen, Harri Lähdesmäki, and Samuel Kaski. Deep learning with differential Gaussian process flows. In *International Conference on Artificial Intelligence and Statistics*, pages 1–15, 2019.

[11] Quoc V Le, Alex J Smola, and Stéphane Canu. Heteroscedastic Gaussian process regression. In *Proceedings of the 22nd international conference on Machine learning*, pages 489–496, 2005.

[12] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International conference on artificial intelligence and statistics*, pages 3870–3882. PMLR, 2020.

[13] Juan Maroñas, Oliver Hamelijnck, Jeremias Knoblauch, and Theodoros Damoulas. Transforming Gaussian processes with normalizing flows. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pages 1081–1089. PMLR, 2021.

[14] Paris Perdikaris, Maziar Raissi, Andreas Damianou, Neil D. Lawrence, and George Em Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A*, 473(2198):20160751, 2017.

[15] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA, 2006.

[16] Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[17] S Vijayakumar and S Schaal. An o(n) algorithm for incremental real time learning in high dimensional space. In *Proc. of the Seventeenth International Conference on Machine Learning*, pages 1079–1086, 2000.

[18] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016.

[19] Kacper Wyrwal, Andreas Krause, and Viacheslav Borovitskiy. Residual deep Gaussian processes on manifolds. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Zheng Zhao, Muhammad Emzir, and Simo Särkkä. Deep state-space Gaussian processes. *Statistics and Computing*, 2021.

A Proof of Proposition 3.1

Recall that the single-sample DGP objective is

$$\hat{\mathcal{L}}_{\mathrm{DGP}}(\boldsymbol{\theta}) = \log p(\mathbf{y} \mid \mathbf{f}^{(L)}) - \sum_{\ell=1}^{L} \mathrm{KL}\big[\cdot \,\|\, \cdot\big].$$

Let L_{DGP} denote a Lipschitz constant of $\hat{\mathcal{L}}_{\mathrm{DGP}}(\boldsymbol{\theta})$. By standard smoothness arguments,

$$L_{\mathrm{DGP}} \le \sup_{\boldsymbol{\theta}} \left\| \nabla_{\boldsymbol{\theta}} \hat{\mathcal{L}}_{\mathrm{DGP}}(\boldsymbol{\theta}) \right\|.$$

Using the decomposition of the objective and the triangle inequality, we obtain

$$L_{\mathrm{DGP}} \le L_{\mathrm{data}} + L_{\mathrm{KL}},$$

where

$$L_{\mathrm{data}} \coloneqq \sup_{\boldsymbol{\theta}} \left\| \nabla_{\boldsymbol{\theta}} \log p(\mathbf{y} \mid \mathbf{f}^{(L)}) \right\|, \qquad L_{\mathrm{KL}} \coloneqq \sup_{\boldsymbol{\theta}} \left\| \nabla_{\boldsymbol{\theta}} \sum_{\ell=1}^{L} \mathrm{KL}\big[\cdot \,\|\, \cdot\big] \right\|.$$

We bound these two terms separately.

Lemma A.1. *Assume that*

$$\left\| \frac{\partial \log p(\mathbf{y} \mid \mathbf{f}^{(L)})}{\partial \mathbf{h}^{(L)}} \right\| \le C_{\mathrm{out}}, \qquad \left\| \frac{\partial f^{(k)}(\mathbf{h}^{(k-1)})}{\partial \boldsymbol{\theta}^{(k)}} \right\| \le C_{\mathrm{param}},$$

for all $k = 1, \dots, L$, and that each sampled layer function is λ_ℓ-Lipschitz with respect to its input, uniformly over $\boldsymbol{\theta}$, i.e.

$$\left\| \frac{\partial f^{(\ell)}(\mathbf{h}^{(\ell-1)})}{\partial \mathbf{h}^{(\ell-1)}} \right\| \le \lambda_\ell, \qquad \ell = 1, \dots, L.$$

Then

$$L_{\mathrm{data}} \le C \left(\sum_{k=1}^{L} \prod_{\ell=k+1}^{L} \lambda_\ell^2 \right)^{1/2},$$

where $C \coloneqq C_{\mathrm{out}} C_{\mathrm{param}}$ is independent of L.

Proof. Write

$$\boldsymbol{\theta} = \big(\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(L)}\big),$$

where $\boldsymbol{\theta}^{(k)}$ is the parameter block associated with layer k. For each $k \in \{1, \dots, L\}$, by the chain rule,

$$\frac{\partial \log p(\mathbf{y} \mid \mathbf{f}^{(L)})}{\partial \boldsymbol{\theta}^{(k)}} = \frac{\partial \log p(\mathbf{y} \mid \mathbf{f}^{(L)})}{\partial \mathbf{h}^{(L)}} \frac{\partial \mathbf{h}^{(L)}}{\partial \boldsymbol{\theta}^{(k)}}.$$

Since the latent states satisfy

$$\mathbf{h}^{(\ell)} = f^{(\ell)}\big(\mathbf{h}^{(\ell-1)}\big) + \boldsymbol{\epsilon}^{(\ell)},$$

and the additive noise $\boldsymbol{\epsilon}^{(\ell)}$ does not depend on $\boldsymbol{\theta}$, we have

$$\frac{\partial \mathbf{h}^{(\ell)}}{\partial \mathbf{h}^{(\ell-1)}} = \frac{\partial f^{(\ell)}(\mathbf{h}^{(\ell-1)})}{\partial \mathbf{h}^{(\ell-1)}}, \qquad \frac{\partial \mathbf{h}^{(k)}}{\partial \boldsymbol{\theta}^{(k)}} = \frac{\partial f^{(k)}(\mathbf{h}^{(k-1)})}{\partial \boldsymbol{\theta}^{(k)}}.$$

Therefore,

$$\frac{\partial \mathbf{h}^{(L)}}{\partial \boldsymbol{\theta}^{(k)}} = \left(\prod_{\ell=k+1}^{L} \frac{\partial \mathbf{h}^{(\ell)}}{\partial \mathbf{h}^{(\ell-1)}} \right) \frac{\partial \mathbf{h}^{(k)}}{\partial \boldsymbol{\theta}^{(k)}} = \left(\prod_{\ell=k+1}^{L} \frac{\partial f^{(\ell)}(\mathbf{h}^{(\ell-1)})}{\partial \mathbf{h}^{(\ell-1)}} \right) \frac{\partial f^{(k)}(\mathbf{h}^{(k-1)})}{\partial \boldsymbol{\theta}^{(k)}}.$$

Taking norms and using sub-multiplicativity of operator norms yields

$$\left\| \frac{\partial \log p(\mathbf{y} \mid \mathbf{f}^{(L)})}{\partial \boldsymbol{\theta}^{(k)}} \right\| \leq \left\| \frac{\partial \log p(\mathbf{y} \mid \mathbf{f}^{(L)})}{\partial \mathbf{h}^{(L)}} \right\| \prod_{\ell=k+1}^{L} \left\| \frac{\partial f^{(\ell)}(\mathbf{h}^{(\ell-1)})}{\partial \mathbf{h}^{(\ell-1)}} \right\| \left\| \frac{\partial f^{(k)}(\mathbf{h}^{(k-1)})}{\partial \boldsymbol{\theta}^{(k)}} \right\|.$$

By the assumptions,

$$\left\| \frac{\partial \log p(\mathbf{y} \mid \mathbf{f}^{(L)})}{\partial \boldsymbol{\theta}^{(k)}} \right\| \leq C_{\text{out}} C_{\text{param}} \prod_{\ell=k+1}^{L} \lambda_\ell.$$

Hence,

$$\left\| \nabla_{\boldsymbol{\theta}} \log p(\mathbf{y} \mid \mathbf{f}^{(L)}) \right\| = \left(\sum_{k=1}^{L} \left\| \nabla_{\boldsymbol{\theta}^{(k)}} \log p(\mathbf{y} \mid \mathbf{f}^{(L)}) \right\|^2 \right)^{1/2} \leq C_{\text{out}} C_{\text{param}} \left(\sum_{k=1}^{L} \prod_{\ell=k+1}^{L} \lambda_\ell^2 \right)^{1/2}.$$

Taking the supremum over $\boldsymbol{\theta}$ gives the result.

□

Lemma A.2. *Under the layerwise variational parameterization of the DGP objective, the KL regularizer decomposes as*

$$\sum_{\ell=1}^{L} \mathrm{KL}[\cdot \,\|\, \cdot] = \sum_{\ell=1}^{L} \mathrm{KL}_\ell(\boldsymbol{\theta}^{(\ell)}).$$

If each layerwise KL term has uniformly bounded gradient, namely

$$\sup_{\boldsymbol{\theta}^{(\ell)}} \left\| \nabla_{\boldsymbol{\theta}^{(\ell)}} \mathrm{KL}_\ell(\boldsymbol{\theta}^{(\ell)}) \right\| \leq C_{\mathrm{KL}}, \qquad \ell = 1, \ldots, L,$$

then

$$L_{\mathrm{KL}}(L) = O(\sqrt{L}).$$

Proof. Since the KL regularizer decomposes layerwise, we have

$$\nabla_{\boldsymbol{\theta}} \sum_{\ell=1}^{L} \mathrm{KL}_\ell(\boldsymbol{\theta}^{(\ell)}) = \big(\nabla_{\boldsymbol{\theta}^{(1)}} \mathrm{KL}_1, \ldots, \nabla_{\boldsymbol{\theta}^{(L)}} \mathrm{KL}_L\big).$$

Therefore,

$$\left\| \nabla_{\boldsymbol{\theta}} \sum_{\ell=1}^{L} \mathrm{KL}_\ell(\boldsymbol{\theta}^{(\ell)}) \right\| = \left(\sum_{\ell=1}^{L} \left\| \nabla_{\boldsymbol{\theta}^{(\ell)}} \mathrm{KL}_\ell(\boldsymbol{\theta}^{(\ell)}) \right\|^2 \right)^{1/2} \leq \left(\sum_{\ell=1}^{L} C_{\mathrm{KL}}^2 \right)^{1/2} = \sqrt{L}\, C_{\mathrm{KL}}.$$

Taking the supremum over $\boldsymbol{\theta}$ yields

$$L_{\mathrm{KL}} \leq \sqrt{L}\, C_{\mathrm{KL}},$$

and hence $L_{\mathrm{KL}}(L) = O(\sqrt{L})$.

□

Combining Lemmas~A.1 and~A.2, we obtain

$$L_{\mathrm{DGP}} \leq C \left(\sum_{k=1}^{L} \prod_{\ell=k+1}^{L} \lambda_\ell^2 \right)^{1/2} + O(\sqrt{L}),$$

where the $O(\sqrt{L})$ term captures the contribution of the KL regularizer. This proves Proposition 3.1.

B Proof of Theorem 4.1

Proof of Theorem 4.1. Define

$$\boldsymbol{\Delta}_t = \mathbf{h}_t^{(1)} - \mathbf{h}_t^{(2)}, \tag{14}$$

so that $\boldsymbol{\Delta}_0 = \mathbf{0}$. Subtracting the two SDEs driven by the same Brownian motion gives

$$d\boldsymbol{\Delta}_t = \big[f(\mathbf{h}_t^{(1)}, t; \boldsymbol{\theta}_1) - f(\mathbf{h}_t^{(2)}, t; \boldsymbol{\theta}_2)\big]\, dt + (\sigma_t - \sigma_t)\, d\mathbf{w}_t. \tag{15}$$

Hence the stochastic terms cancel exactly and

$$\boldsymbol{\Delta}_t = \int_0^t \big[f(\mathbf{h}_s^{(1)}, s; \boldsymbol{\theta}_1) - f(\mathbf{h}_s^{(2)}, s; \boldsymbol{\theta}_2)\big]\, ds. \tag{16}$$

Adding and subtracting $f(\mathbf{h}_s^{(2)}, s; \boldsymbol{\theta}_1)$ yields

$$\boldsymbol{\Delta}_t = \int_0^t \big[f(\mathbf{h}_s^{(1)}, s; \boldsymbol{\theta}_1) - f(\mathbf{h}_s^{(2)}, s; \boldsymbol{\theta}_1)\big] + \big[f(\mathbf{h}_s^{(2)}, s; \boldsymbol{\theta}_1) - f(\mathbf{h}_s^{(2)}, s; \boldsymbol{\theta}_2)\big] ds. \tag{17}$$

Taking norms and using the Lipschitz assumptions,

$$\big\| f(\mathbf{h}_s^{(1)}, s; \boldsymbol{\theta}_1) - f(\mathbf{h}_s^{(2)}, s; \boldsymbol{\theta}_1) \big\| \leq L_h \|\boldsymbol{\Delta}_s\|,$$
$$\big\| f(\mathbf{h}_s^{(2)}, s; \boldsymbol{\theta}_1) - f(\mathbf{h}_s^{(2)}, s; \boldsymbol{\theta}_2) \big\| \leq L_\theta \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|.$$

Therefore, with $u(t) = \|\boldsymbol{\Delta}_t\|$ and $\delta_\theta = \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|$,

$$u(t) \leq \int_0^t \left(L_h u(s) + L_\theta \delta_\theta\right)\, ds. \tag{18}$$

Define

$$y(t) = \int_0^t (L_h u(s) + L_\theta \delta_\theta)\, ds, \tag{19}$$

so that $u(t) \le y(t)$ and $y(0) = 0$. Then

$$\begin{aligned} \frac{dy(t)}{dt} &= L_h u(t) + L_\theta \delta_\theta \\ &\le L_h y(t) + L_\theta \delta_\theta. \end{aligned} \tag{20}$$

Multiplying by $e^{-L_h t}$ gives

$$\frac{d}{dt}\left[e^{-L_h t} y(t)\right] \le L_\theta e^{-L_h t} \delta_\theta. \tag{21}$$

Integrating from 0 to t and using $y(0) = 0$, we obtain, for $L_h > 0$,

$$e^{-L_h t} y(t) \le L_\theta \delta_\theta \int_0^t e^{-L_h s}\, ds = \frac{L_\theta \delta_\theta}{L_h}(1 - e^{-L_h t}), \tag{22}$$

and hence

$$y(t) \le \frac{L_\theta}{L_h}(e^{L_h t} - 1)\delta_\theta. \tag{23}$$

If $L_h = 0$, Eq. (18) reduces to

$$u(t) \le \int_0^t L_\theta \delta_\theta\, ds = L_\theta t\, \delta_\theta. \tag{24}$$

Combining the two cases,

$$\|\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(2)}\| \le L_\theta\, \phi(L_h, t)\, \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|, \tag{25}$$

where

$$\phi(L_h, t) = \begin{cases} \dfrac{e^{L_h t} - 1}{L_h}, & L_h > 0, \\ t, & L_h = 0. \end{cases} \tag{26}$$

□

C Proof of Corollary 4.2

Proof. Starting from the pathwise ODE for $\boldsymbol{\Delta}_t$ established in the proof of Theorem 4.1, we have

$$\frac{1}{2}\frac{d}{dt}\|\boldsymbol{\Delta}_t\|^2 = \left\langle \boldsymbol{\Delta}_t,\, f(\mathbf{h}_t^{(1)}, t; \boldsymbol{\theta}_1) - f(\mathbf{h}_t^{(2)}, t; \boldsymbol{\theta}_2) \right\rangle.$$

Adding and subtracting $f(\mathbf{h}_t^{(2)}, t; \boldsymbol{\theta}_1)$ yields

$$\frac{1}{2}\frac{d}{dt}\|\boldsymbol{\Delta}_t\|^2 = \underbrace{\left\langle \boldsymbol{\Delta}_t,\, f(\mathbf{h}_t^{(1)}, t; \boldsymbol{\theta}_1) - f(\mathbf{h}_t^{(2)}, t; \boldsymbol{\theta}_1) \right\rangle}_{\text{(I)}} + \underbrace{\left\langle \boldsymbol{\Delta}_t,\, f(\mathbf{h}_t^{(2)}, t; \boldsymbol{\theta}_1) - f(\mathbf{h}_t^{(2)}, t; \boldsymbol{\theta}_2) \right\rangle}_{\text{(II)}}.$$

By the one-sided contractivity assumption, (I) $\le -\mu\|\boldsymbol{\Delta}_t\|^2$. For (II), Cauchy–Schwarz and the Lipschitz condition in $\boldsymbol{\theta}$ give

$$\text{(II)} \le \|\boldsymbol{\Delta}_t\|\, L_\theta \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|.$$

Applying Young's inequality with parameter μ,

$$\|\boldsymbol{\Delta}_t\| \, L_\theta \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| \leq \frac{\mu}{2}\|\boldsymbol{\Delta}_t\|^2 + \frac{L_\theta^2}{2\mu}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|^2,$$

and combining the two bounds yields

$$\frac{d}{dt}\|\boldsymbol{\Delta}_t\|^2 \leq -\mu\|\boldsymbol{\Delta}_t\|^2 + \frac{L_\theta^2}{\mu}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|^2.$$

Since $\boldsymbol{\Delta}_0 = \mathbf{0}$, Grönwall's inequality gives

$$\|\boldsymbol{\Delta}_t\|^2 \leq \frac{L_\theta^2}{\mu^2}\big(1 - e^{-\mu t}\big)\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|^2.$$

Taking square roots and evaluating at $t = T$,

$$\|\mathbf{h}_T^{(1)} - \mathbf{h}_T^{(2)}\| \leq \frac{L_\theta}{\mu}\sqrt{1 - e^{-\mu T}}\,\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| \leq \frac{L_\theta}{\mu}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|,$$

for almost every sample path.

□

D Complexity Analysis

In CDGP, the inducing set is the Cartesian product $\mathbf{Z}_h \times \mathbf{Z}_t$ of size $M_h M_t$. Exploiting the Kronecker structure of the separable kernel, the per-iteration complexity is $\mathcal{O}(M_h^3 + M_t^3 + NJM_hM_t)$.

In FlowDGP, inference is performed only over the base GP f_0 with M inducing points, giving a per-iteration complexity of $\mathcal{O}(M^3 + NM^2 + NJC_g)$, where C_g is the cost of evaluating g_ϕ. The reduction from M_hM_t to M inducing points more than compensates for the additional ODE solver cost, resulting in substantially faster training in practice.

This complexity gap is reflected in the benchmark train-time results in Table 1, where FlowDGP is already substantially faster than CDGP on the smaller benchmark subsets. That empirical gap is the main reason why CDGP and residual DGP are not carried into the full-size large-scale experiment.

E Simulation Details

The experiments were run on a single NVIDIA RTX 4500 Ada Generation GPU (24 GB). Note that, for each test input $\mathbf{x}_n$, we draw $S = 300$ Monte Carlo samples $\{f_s(\mathbf{x}_n)\}_{s=1}^S$ from the model and form a Gaussian predictive distribution $\mathcal{N}(\mu_n, \sigma_n^2)$ whose moments are estimated from the samples:

$$\mu_n = \frac{1}{S}\sum_{s=1}^{S} f_s(\mathbf{x}_n), \qquad \sigma_n^2 = \frac{1}{S}\sum_{s=1}^{S}\big(f_s(\mathbf{x}_n) - \mu_n\big)^2 + \sigma_{\text{noise}}^2. \tag{27}$$

The negative log-likelihood is then evaluated as

$$\text{NLL} = \frac{1}{2N}\sum_{n=1}^{N}\left[\log(2\pi\sigma_n^2) + \frac{(y_n - \mu_n)^2}{\sigma_n^2}\right]. \tag{28}$$

E.1 Datasets Description

We evaluate on a one-dimensional toy example, three synthetic benchmarks, three real-world regression tasks at two sample sizes, and YearPredictionMSD. For the toy and benchmark datasets, inputs are standardised per dimension and targets are standardised by the training-set mean and standard deviation. The toy example is used in Section 5.1 with $N_{\text{train}} = 300$ and $N_{\text{test}} = 500$. For the benchmark experiments (Section 5.2), each dataset uses $N_{\text{train}} = 300$ and $N_{\text{test}} = 500$; for the real-world tasks, we randomly subsample from the full datasets to this size. For the large-scale experiments (Section 5.3), the full versions of the real-world datasets are used.

1D toy example (`toy_example`). Inputs $x \in [0, 1]$ are drawn uniformly, and the scalar target is $y = f(x) + \sigma(x)\,\varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 1)$. Writing $\tilde{x} = (x - x_{\min})/(x_{\max} - x_{\min})$ for the normalized input, the clean signal is a superposition of seven components, $f(x) = \sum_{i=1}^{7} f_i(\tilde{x})$, where

$$
\begin{aligned}
f_1(\tilde{x}) &= \big[0.08 + 0.42\,\mathrm{sigmoid}(22(\tilde{x} - 0.28)) + 0.82\,\tilde{x}^2\big]\,\sin\phi(\tilde{x}),\\
f_2(\tilde{x}) &= 0.12\sin(2\phi(\tilde{x}) + 6\tilde{x}) + 0.10\sin(92\,\tilde{x}^2)\\
&\quad + 0.08\sin\big(36\tilde{x} + 2.2\sin(4\pi\tilde{x})\big) + 0.10(\tilde{x} - 0.48)^2,\\
f_3(\tilde{x}) &= 0.42\sin(2.2\pi\tilde{x}) + 0.38\sin(56\pi\tilde{x}),\\
f_4(\tilde{x}) &= 0.32\,\exp\big(-\tfrac{1}{2}(\tilde{x} - 0.5)^2/0.11^2\big)\sin(84\pi\tilde{x}),\\
f_5(\tilde{x}) &= 0.28\sqrt{|\tilde{x} - 0.34|},\\
f_6(\tilde{x}) &= 0.45\,\mathrm{sigmoid}\big(110(\tilde{x} - 0.58)\big),\\
f_7(\tilde{x}) &= 0.32\big[w(\tilde{x})\,\sin\big(9\tilde{x}(0.65 + 0.35\tilde{x})\big)\\
&\quad + (1 - w(\tilde{x}))\sin(52\tilde{x})\big],
\end{aligned}
$$

with chirp phase $\phi(\tilde{x}) = \frac{\omega_{\text{hi}}}{\beta}\big(1 - e^{-\beta\tilde{x}}\big) + \omega_{\text{lo}}\,\tilde{x}$ (parameters $\omega_{\text{hi}} = 88$, $\omega_{\text{lo}} = 5$, $\beta = 9.5$) and regime-switching gate $w(\tilde{x}) = \mathrm{sigmoid}(28(\tilde{x} - 0.45))$. The seven components respectively contribute: (1) a chirp with growing envelope, (2) second-harmonic, quadratic-phase, and FM terms with a mild polynomial trend, (3) spectrally conflicting slow and fast sinusoids, (4) a localised high-frequency burst inside a narrow Gaussian window, (5) a square-root cusp, (6) a near-discontinuous step, and (7) a regime blend whose dominant frequency differs sharply between the two sides of the domain. Noise is heteroscedastic with $\sigma(x) = 0.16\big(0.55 + 0.9\,\tilde{x}\big)$, so the output variance grows roughly $2.6\times$ from left to right. We use $N_{\text{train}} = 300$ and $N_{\text{test}} = 500$; inputs and targets are standardised using training-set statistics. Although the task is strictly one-dimensional ($D_x = 1$), the combination of multi-scale sinusoids, a localised burst, a cusp, a near-step, and a regime change makes it adversarial to any model with a single stationary lengthscale.

Heteroscedastic (`heteroscedastic`). Inputs $\mathbf{x} \in \mathbb{R}^{D_x}$ with $D_x = 10$ are drawn uniformly on $[-4, 4]^{D_x}$. The latent signal is

$$
g(\mathbf{x}) = \sin(2x_1) + 0.5\cos(x_2) + 0.3x_3, \qquad f_k(\mathbf{x}) = g(\mathbf{x}) + 0.2\sin(kx_1), \quad k = 1, \ldots, 10,
$$

and intermediate observations are $\tilde{y}_k = f_k(\mathbf{x}) + \sigma(\mathbf{x})\,\varepsilon_k$ with $\varepsilon_k \sim \mathcal{N}(0, 1)$ and input-dependent noise scale

$$
\sigma(\mathbf{x}) = 0.1 + 0.5\,\mathrm{sigmoid}(1.5x_1 - 0.5x_2).
$$

The scalar target is $y = \frac{1}{10}\sum_{k=1}^{10} \tilde{y}_k$, so the effective noise is heteroscedastic with roughly a $5\times$ variance ratio across the input space. This benchmark evaluates a model's ability to fit input-dependent aleatoric uncertainty.

Nonstationary (`nonstationary`). Inputs $\mathbf{x} \in \mathbb{R}^{D_x}$ with $D_x = 10$ are drawn uniformly on $[-4, 4]^{D_x}$. The clean signal is a smooth two-regime blend governed by the first coordinate,

$$
f(\mathbf{x}) = \big(1 - w(x_1)\big)\sin(x_1) + w(x_1)\cos(x_1) + 0.2\,x_2, \qquad w(x_1) = \mathrm{sigmoid}(x_1),
$$

with homoscedastic Gaussian noise $\varepsilon \sim \mathcal{N}(0, 0.2^2)$. The change in effective lengthscale between the $x_1 \ll 0$ and $x_1 \gg 0$ regimes is the principal challenge: a single stationary kernel cannot match both regimes simultaneously.

Heavy-tailed (`heavy_tailed`). Inputs $\mathbf{x} \in \mathbb{R}^{D_x}$ with $D_x = 10$ are drawn uniformly on $[-5, 5]^{D_x}$, and the latent signal takes the additive form

$$f(\mathbf{x}) = \frac{1}{D_x} \sum_{i=1}^{D_x} \big(\sin(1.5x_i) + 0.3x_i\big).$$

Observations are $y = f(\mathbf{x}) + \varepsilon$ with Student-t noise $\varepsilon \sim t_\nu(0, s)$, where $\nu = 3$ and $s = 0.3$. The fat tails induce a non-negligible fraction of gross outliers, which stresses baselines that assume Gaussian noise.

SARCOS (`sarcos`, `sarcos_small`). The SARCOS inverse-dynamics benchmark [17]: given $D_x = 21$ joint-state features (seven positions, seven velocities, seven accelerations) from a seven-degree-of-freedom anthropomorphic robot arm, predict the first joint torque. We use the standard split with $44{,}484$ training and $4{,}449$ test samples in the large-scale setting, and subsample to $N_{\text{train}} = 300$ and $N_{\text{test}} = 500$ in the benchmark setting (`sarcos_small`). Features and targets are standardised using the full parent training set.

UCI Bike (`uci_bike`, `uci_bike_small`). The UCI Bike Sharing dataset [8]: the target is the hourly rental count (`cnt`), and the inputs are $D_x = 12$ weather and calendar features (`season`, `yr`, `mnth`, `hr`, `holiday`, `weekday`, `workingday`, `weathersit`, `temp`, `atemp`, `hum`, `windspeed`). We apply an $80/20$ time-ordered split with per-feature z-scoring and target standardisation, yielding $13{,}903$ training and $3{,}476$ test samples in the large-scale setting; the benchmark setting subsamples to $N_{\text{train}} = 300$ and $N_{\text{test}} = 500$. The target is highly non-Gaussian (count data with mixed diurnal and seasonal regimes), making this a useful test of non-stationarity.

UCI Energy (`uci_energy`, `uci_energy_small`). The UCI Appliances Energy Prediction dataset [2]: the target is appliance energy consumption (Wh), and the inputs are $D_x = 29$ indoor sensor readings and weather variables, augmented with four derived temporal features (hour of day, day of week, month, weekend indicator) parsed from the timestamp. Metadata columns (`date`, `rv1`, `rv2`) and the target itself are excluded from the inputs. We apply a chronological $80/20$ split with z-scoring, yielding $15{,}788$ training and $3{,}947$ test samples in the large-scale setting; the benchmark setting subsamples to $N_{\text{train}} = 300$ and $N_{\text{test}} = 500$. Targets are strongly right-skewed with sparse high-activity spikes, producing effectively heavy-tailed residuals in small samples.

YearPredictionMSD. YearPredictionMSD [1] is a song-release-year prediction dataset containing $515{,}345$ observations, each represented by 90 numerical audio features. The task is scalar regression from audio characteristics to release year.

Summary. Table 2 summarises the input dimension, sample sizes (benchmark setting), and principal regression challenge targeted by each dataset. All tasks are scalar regression ($D_y = 1$); sample sizes for the corresponding large-scale datasets (`sarcos`, `uci_bike`, `uci_energy`) are given in the paragraphs above.

E.2 Hyperparameter Settings

To ensure reproducibility, we provide the hyperparameter configurations for all methods. Table 3 lists the hyperparameters selected via grid search, and Table 4 reports the fixed settings shared across experiments. For each method and dataset, hyperparameters were selected based on validation performance, considering both NLL and RMSE. We first identified configurations whose validation NLL was close to the best-performing one, and among them chose the configuration with the lowest validation RMSE. This favors models that achieve strong probabilistic performance without sacrificing point prediction accuracy. In these tables, J is the number of integration steps and M_t is the number of inducing locations along depth for CDGP. Neither denotes the terminal horizon T, which is 1.0 for both CDGP and FlowDGP unless varied explicitly.

Dataset	D_x	N_{train}	N_{test}	Primary challenge
`heteroscedastic`	10	300	500	input-dependent noise
`nonstationary`	10	300	500	two-regime lengthscale change
`heavy_tailed`	10	300	500	Student-t ($\nu = 3$) outliers
`sarcos_small`	21	300	500	high-dimensional nonlinear dynamics
`uci_bike_small`	12	300	500	mixed diurnal/seasonal regimes
`uci_energy_small`	29	300	500	skewed, spiky real sensor data
`sarcos`	21	44,484	4,449	high-dimensional nonlinear dynamics
`uci_bike`	12	13,903	3,476	mixed diurnal/seasonal regimes
`uci_energy`	29	15,788	3,947	skewed, spiky real sensor data

Table 2: Summary of evaluation datasets. The first block lists the three synthetic benchmarks, the second block the subsampled real-world datasets used in Section 5.2, and the third block their full-size counterparts used in Section 5.3. All tasks are scalar regression with $D_y = 1$.

Table 3: Grid-searched hyperparameters for the benchmark methods. For each method and dataset, we select the configuration with the lowest validation NLL.

Method	**Search space**
SparseGP	`lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
HeteroskedasticGP	`lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
DKL	`grid_size` $\in \{80, 100, 120\}$ `num_dims` $\in \{2, 3\}$ `lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
TDGP	`m_v` $\in \{20, 30\}$ `m_u` $\in \{20, 30\}$ `lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
DGP	`num_layers` $\in \{2, 3, 5\}$ `lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
ResidualDGP	`num_layers` $\in \{2, 3, 5\}$ `lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
NODE	`hidden_dim` $\in \{32, 64\}$ `num_layers` $\in \{1, 2\}$ $J \in \{2, 3, 5\}$ `lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
NSDE	`hidden_dim` $\in \{32, 64\}$ $J \in \{2, 3, 5\}$ `lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
CDGP	$J \in \{2, 3, 5\}$ $M_t \in \{3, 5\}$ `lr` $\in \{0.01, 0.02, 0.03, 0.05\}$
FlowDGP	`hidden_dims` $\in \{[32, 32], [64, 64]\}$ $J \in \{2, 3, 5\}$ `lr` $\in \{0.01, 0.02, 0.03, 0.05\}$

F Additional Experimental Comparisons

F.1 Large-scale comparison on YearPredictionMSD

We compare FlowDGP with DKL, NODE and NSDE on YearPredictionMSD (Appendix E.1). Table 5 reports the results: FlowDGP achieves the lowest mean RMSE and NLL, while NSDE achieves the lowest mean CRPS.

Table 4: Fixed hyperparameter settings for the toy and benchmark experiments. Method-specific entries override the defaults.

Scope	Setting
Defaults	`num_inducing` $= 30$ (whenever applicable) `epochs` $= 2500$ optimiser $=$ Adam
DKL	`epochs` $= 500$
CDGP	$\sigma = 0.15$ $T = 1.0$ (unless otherwise specified)
FlowDGP	$\sigma = 0.15$ $T = 1.0$ (unless otherwise specified)
NODE	$T = 1.0$ `solver` $=$ Euler `rtol` $= 10^{-4}$, `atol` $= 10^{-5}$ `adjoint` $=$ False
NSDE	$T = 1.0$ `solver` $=$ Euler `activation` $=$ LipSwish
TDGP	`diag_qmu` $=$ True `epochs=1000`
TGP	`num_blocks` $= 3$ (SAL) `hidden_dims` $= [50, 50]$ `epochs` $= 15000$ `lr` $= 0.001$

Table 5: YearPredictionMSD results, mean $\pm$ standard deviation over random seeds. Lower is better; bold marks the lowest mean.

Method	RMSE	NLL	CRPS
DKL	0.8090 ± 0.0022	1.2078 ± 0.0023	0.4243 ± 0.0012
NODE	0.8096 ± 0.0018	1.2088 ± 0.0025	0.4252 ± 0.0018
NSDE	0.8075 ± 0.0022	1.2074 ± 0.0033	$\mathbf{0.4227 \pm 0.0016}$
FlowDGP	$\mathbf{0.8052 \pm 0.0015}$	$\mathbf{1.2027 \pm 0.0023}$	0.4234 ± 0.0016

F.2 A controlled contraction illustration

To illustrate Corollary 4.2, we consider an auxiliary one-dimensional SDE, separate from the fitted CDGP and FlowDGP models. Paired paths share the same Brownian motion, and we measure

$$S(T) = \frac{|h_T(\theta_2) - h_T(\theta_1)|}{|\theta_2 - \theta_1|}.$$

The contractive drift is $f(h;\theta) = -\mu h - 0.5\tanh(h) + \theta$; the control is $f(h;\theta) = 0.25h + \theta$. For the contractive drift, $\partial_h f = -\mu - 0.5\,\mathrm{sech}^2(h) \leq -\mu$ and $L_\theta = 1$. With equal initial conditions and shared parameter-independent additive noise, the corollary bounds sensitivity by $1/\mu$. Table 6 reports the results from 5,000 paired paths. Sensitivity levels off below the bound in the contractive cases, while the control increases with the horizon.

Table 6: Controlled SDE illustration of Corollary 4.2.

Drift	μ	$S(1)$	$S(4)$	$S(8)$	L_θ/μ
Contractive	0.5	0.637	1.002	1.023	2.000
Contractive	1.0	0.521	0.672	0.673	1.000
Contractive	2.0	0.368	0.401	0.401	0.500
Non-contractive	—	1.136	6.870	25.542	—

F.3 Learned drift evolution

We use the toy dataset to visualize how the learned drift and latent state evolve with continuous depth in CDGP and FlowDGP.

Both models use $T = 1$, $J = 3$, 30 inducing points for each state or base GP, and 2,500 training epochs; CDGP uses five depth inducing points. The FlowDGP drift posterior mean is estimated using 512 base-GP samples. We plot eight latent trajectories for each of seven inputs.

Figure 3 plots the drift posterior means at fixed states for CDGP and fixed inputs for FlowDGP, together with sampled latent trajectories. In this toy example, CDGP's drift means change modestly with t, whereas FlowDGP's vary more substantially. The latent trajectories show a similar pattern, with more pronounced changes across depth for FlowDGP.

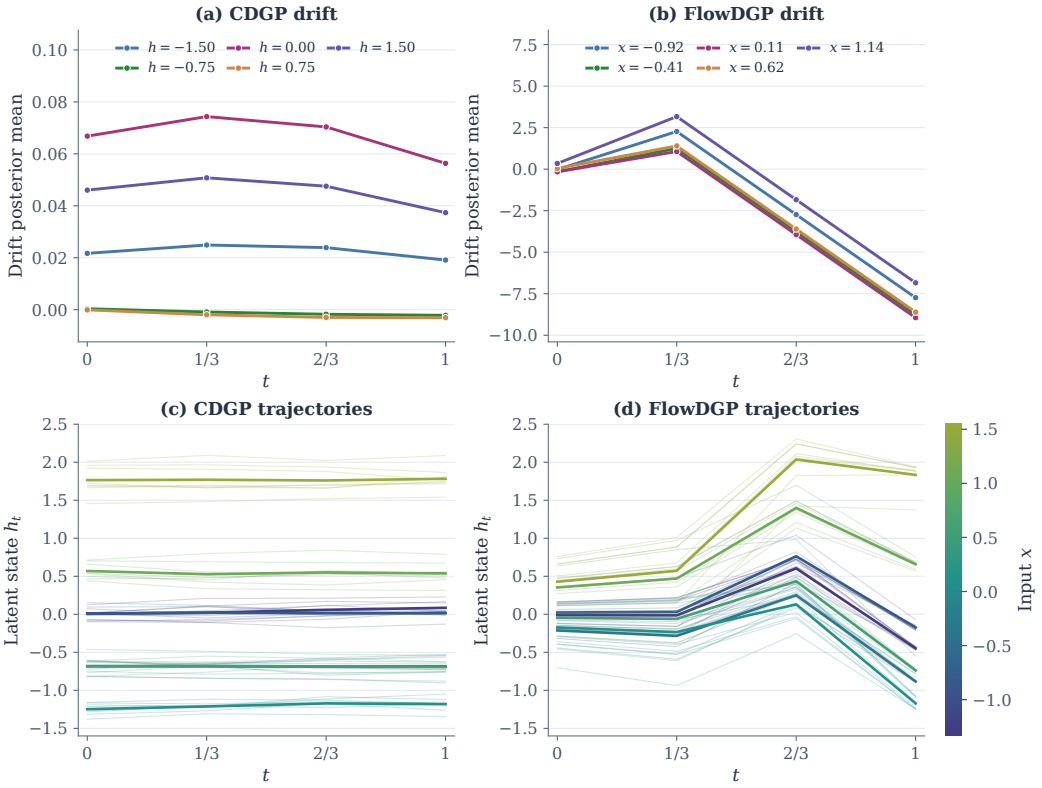


Figure 3: **Learned drift evolution and latent trajectories on the toy dataset.** Top: drift posterior means versus t at fixed states h for CDGP and fixed inputs x for FlowDGP. Bottom: sampled latent trajectories (thin lines) and their means (thick lines), colored by input.

G LLM Usage

Large language models were used solely for editing and polishing the writing of this paper.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's four contributions: (i) a continuous-depth SDE methodology for DGP regression with a GP-based prior over the depth-indexed drift; (ii) two concrete drift-prior instantiations, CDGP and FlowDGP, within a shared sparse variational framework; (iii) an analysis identifying the repeated Jacobian products that control sensitivity in discrete DGPs and a finite-horizon pathwise sensitivity bound for the continuous-depth formulation; and (iv) empirical validation that the previously proposed Residual DGP is a meaningful first repair of plain DGP, while CDGP and FlowDGP remain stable or competitive across the studied regression regimes. The stronger contractive corollary is a specialized auxiliary result and was not used to design CDGP or FlowDGP. We do not claim Residual DGP as a new method. These contributions are developed in Sections 3 and 4, and supported by the experiments in Sections 5.1, 5.2, and 5.3.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Section 8, including the focus on moderate-dimensional regression and the open question of how the choice of kernel and flow parameterization affects CDGP and FlowDGP across different data regimes. The restrictiveness of the separable kernel in CDGP is also explicitly discussed in Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Proposition 3.1 and Theorem 4.1 are stated with explicit assumptions in Sections 3 and 4.4, with full proofs provided in Appendices A and B. Supporting Lemmas A.1 and A.2 are included with complete proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Dataset construction is described in Appendix E.1, hyperparameter search spaces and fixed settings are reported in Appendix E.2 (Tables 3 and 4), and the training procedure including the variational objective and Euler–Maruyama discretization is fully specified in Section 4.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All datasets used are publicly available, including UCI Bike Sharing [8], UCI Appliances Energy [2], and SARCOS [17]. The synthetic benchmarks are fully specified in Appendix E.1 with explicit generative formulas, and code is provided as anonymized supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and test splits, optimizer (Adam), epoch counts, learning rate grids, and method specific hyperparameters are detailed in Appendix E.2 (Tables 3 and 4), with dataset specific splits described in Appendix E.1.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Benchmark comparisons, YearPredictionMSD (Table 5), and TGP (Table 1) report means and standard deviations across random seeds. The controlled SDE illustration uses 5,000 paired paths.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources are reported at the start of Appendix E. All experiments were run on a single NVIDIA RTX 4500 Ada Generation GPU (24 GB), and per method training times are reported in Table 1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conforms to the NeurIPS Code of Ethics. It uses publicly available datasets and does not involve human subjects, sensitive data, or applications with foreseeable ethical concerns.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work is a foundational methodological contribution to Gaussian process regression and is not tied to any specific application or deployment with identifiable societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release pretrained models, scraped datasets, or generative systems with risk of misuse. It only proposes a probabilistic regression methodology evaluated on standard public benchmarks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third party datasets (UCI Bike Sharing [8], UCI Appliances Energy [2], and SARCOS [17]) and baseline methods (neural ODE [3], neural SDE [12], DKL [18], TDGP [5], residual DGP [19]) are properly cited in the references, and the datasets are used under their respective public licenses on the UCI repository.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets or pretrained models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Large language models were used solely for editing and polishing the writing of this paper, which does not require declaration per the NeurIPS policy.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.