

A Streaming Sparse Cholesky Method for Derivative-Informed Gaussian Process Surrogates Within Digital Twin Applications

Krishna Prasath Logakannan^a, Shridhar Vashishtha^{b,c}, Jacob Hochhalter^a, Shandian Zhe^b,
Robert M. Kirby^{b,c}

^a*Department of Mechanical Engineering, University of Utah, , Salt Lake City, 84112, UT, USA*

^b*Kahlert School of Computing, University of Utah, Salt Lake City, 84112, UT, USA*

^c*Scientific Computing & Imaging Institute, University of Utah, , Salt Lake City, 84112, UT, USA*

Abstract

Digital twins are developed to model the behavior of a specific physical asset (or twin), and they can consist of high-fidelity physics-based models or surrogates. A highly accurate surrogate is often preferred over multi-physics models as they enable forecasting the physical twin future state in real-time. To adapt to a specific physical twin, the digital twin model must be updated using in-service data from that physical twin. Here, we extend Gaussian process (GP) models to include derivative data, for improved accuracy, with dynamic updating to ingest physical twin data during service. Including derivative data, however, comes at a prohibitive cost of increased covariance matrix dimension. We circumvent this issue by using a sparse GP approximation, for which we develop extensions to incorporate derivatives. Numerical experiments demonstrate that the prediction accuracy of the derivative-enhanced sparse GP method produces improved models upon dynamic data additions. Lastly, we apply the developed algorithm within a DT framework to model fatigue crack growth in an aerospace vehicle.

Keywords: Digital twin, crack growth, sparse GP, derivative-informed GP, dynamic update

1. Introduction

Coining of the term digital twin (DT) is often attributed to the US Air Force and NASA in the early 2000s as a means of creating a dynamic, high-fidelity digital framework to monitor, simulate, and predict a specific physical component or system [1, 2]. Initially, the DT concept was motivated by the need to improve structural reliability estimates during the operational life of aerospace vehicles. The DT concept has since been extended to manufacturing, automation, energy and utilities, healthcare, etc., where physical components or systems would benefit from continuously updated assessments or optimization during their life cycle [3]. Independent of the particular application, the fundamental objective of a DT model is to improve the accuracy of predictions and reduce uncertainty. This is accomplished by incorporating measurable characteristics of an individual physical asset prior to and during its operational life, as opposed to using a nominal or purely stochastic modeling approach.

In an attempt to remain up-to-date with its physical twin (PT), a DT framework often employs surrogate modeling methods [4]. A central challenge of a DT model, therefore, lies in balancing fidelity with computational efficiency: full-scale physics-based simulations, which in and of themselves are surrogates but ones in which we try to ‘throw away’ as little as possible,

would often exceed real-time modeling constraints, while reduced-order models risk losing critical individualized accuracy. Once a DT modeling approach is identified, with an appropriate balance of fidelity and efficiency, an initial DT model can be generated (trained) using, *e.g.*, as-manufactured PT characteristics [5], nominal material information used during the design stage, or nominal data from preceding PTs. While capturing both known (measurable) and unknown (nominal or stochastic) individual characteristics is a critical starting point, this only marks the initialization of the DT modeling process, as outlined in black in Figure 1.

As with the study of biological twins — where identical genetic starting material does not guarantee identical outcomes — DTs must account for external perturbations, operating environments, and system variability that act on the PT during its life. To this end, the DT model and its state must be updated periodically to reflect the measured in-service usage and evolution of the PT. This requisite co-evolution occurs through structural modifications to the DT model form and parameter recalibration as informed by new data, as outlined in red in Figure 1. In this step, there exists an implicit assumption that the DT and PT are not only co-evolving, but also that smoothness in time of the PT is mimicked in the DT. Hence, the DT must not only be responsive but also smoothly adaptive. Lastly, a two-way communication is necessitated in which the DT acquires measured PT usage and state, which are in turn used to generate updated predictions for the PT and used as a basis of decision making (*e.g.*, for maintenance or replacement), as outlined in green in Figure 1. The integration of all these requirements and objectives defines both the promise and the frontier of DT research.

The general DT concept is abstract, so any quantitative assessment requires an application problem. Here, we select the original motivating application: aircraft structural life prediction [1]. The objective of this application problem is to use a DT to establish a condition-based maintenance schedule, as opposed to current conservative methods of scheduled maintenance that are based entirely on flight hours but not individual usage or state, *e.g.*, safe life or damage tolerance approaches [6, 7]. Establishing a reliable DT in this application has direct implications for improved reliability, safety, and cost-effectiveness: an accurate DT model can reduce the frequency of unnecessary maintenance interventions (where issues can be introduced), thereby minimizing system downtime and associated costs. Since the seminal study presented by Tuegel *et al.* [1], which focused on the use of “ultrahigh fidelity models,” research has extended to simulated demonstrations along with the incorporation of surrogate modeling and uncertainty quantification. Liao *et al.* [8] assessed the shift from deterministic individual aircraft tracking to a probabilistic, flight-by-flight approach using Bayesian updating and probabilistic fatigue crack

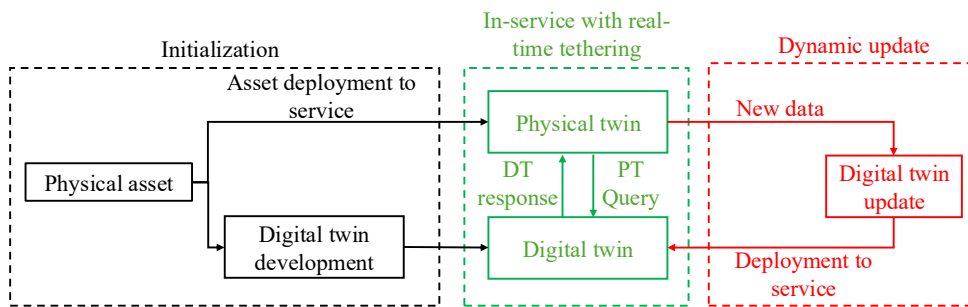


Figure 1: Schematic of workflow of a typical DT system with real-time tethering and dynamic update capability.

growth models. They demonstrated that DTs can reduce conservatism in maintenance planning, improve fatigue life predictions, and enable risk-based decision-making tailored to individual aircraft. Millwater *et al.* [9] presented probabilistic methods for risk assessment in DT frameworks, detailing computational strategies for estimating failure probabilities, remaining useful life, and inspection intervals, highlighting the importance of real-time data integration and surrogate modeling. Because fatigue cracking in aerospace materials is inherently a multi-scale problem, several researchers have employed multi-scale DT modeling strategies. As an example, Whelan and McDowell [10] model microstructure-sensitive fatigue behavior in Ti-6Al-4V. Their study highlights quantifying epistemic uncertainty from model form and parameters, using statistical volume elements (SVEs) and fatigue indicator parameters (FIPs) to assess fatigue resistance. Yeratapally *et al.* and Leser *et al.* present a two-part DT feasibility study that establishes a probabilistic, multi-scale framework for fatigue life prediction in Aluminum alloy 7075-T651. In Part I, Yeratapally *et al.* [11] develop a DT approach that couples microstructurally small crack modeling—using crystal plasticity finite element analysis of SVEs with microstructurally large crack (MLC) modeling via linear elastic fracture mechanics, calibrated through Monte Carlo and Markov Chain Monte Carlo methods. In Part II, Leser *et al.* [12] integrate in-situ diagnostics using digital image correlation and Bayesian inference to iteratively calibrate the DT model as damage becomes observable, significantly reducing uncertainty in prognostic predictions. Within this application domain, past research has largely focused on Bayesian methods for updating DT model parameters, given new observations during service life, while investigations into the structural modifications to the DT model form remain open.

In this work, we develop and employ derivative-informed sparse Gaussian processes (GPs), which are particularly suited for DT applications. These models offer the dual advantages of high predictive accuracy and the ability to incorporate new data dynamically at minimal computational cost, making them well-suited for real-time updating. We present work that advances DT modeling by extending GP surrogates in three key directions. First, we incorporate derivative information into the GP formulation, enabling the surrogate to more effectively capture local sensitivities and improve predictive fidelity, particularly in high-dimensional and multi-physics contexts. Second, we generalize sparse Cholesky factorization methods to handle derivative-enhanced covariance structures, thereby preserving computational tractability even as model complexity increases. This extension allows the surrogate to scale to large datasets while retaining the accuracy benefits of derivative augmentation. Finally, we develop a dynamic update algorithm tailored for the digital twin setting, in which new sensor data and operational measurements arrive sequentially. This algorithm ensures that the GP surrogate can be updated efficiently in real time without retraining from scratch, allowing the DT to evolve in tandem with its physical counterpart. Collectively, these contributions provide a principled and computationally efficient foundation for deploying adaptive, derivative-informed GP surrogates in digital twin applications, with particular relevance to aerospace and mechanical systems where both accuracy and scalability are critical.

The paper is organized as follows. In Section 2, we present a mathematical description of GP modeling and how it can be updated to incorporate derivative information. We highlight the increased accuracy capabilities of our derivative-enhanced GP when used to model functions that are sufficiently smooth. In Section 3, we summarize a previously published sparse Cholesky algorithm with provable properties for solving a GP system [13] and present our modifications of this algorithm to account for derivative-enhanced GP. Given our interest in using derivative-enhanced GPs in a DT context, we present in Section 4 an extension of the sparse Cholesky algorithm applicable in the dynamic (streaming) context. We present several variants

of the algorithm and demonstrate which works best for DT problems. In Section 5, we combine our contributions and show how derivative-enhanced GPs solved using our dynamic sparse Cholesky algorithm can be used to solve a real-world DT application. We summarize our work and document conclusions in Section 6.

2. GP Surrogate Modeling for DT Applications

We present in Section 2.1 a mathematical review of GPs as it is used for surrogate modeling. In Section 2.2, the generalized case of GP modeling with noisy data is presented with our extension of GPs to utilize both function values and their derivatives up to any arbitrary order d . In Section 2.3, we present the specific case of modeling noise-free data within our contribution of derivative-enhanced GPs. In Section 2.4, we present numerical verification tests to demonstrate the correctness of our algorithm and to highlight the superior convergence properties of derivative-enhanced GPs when applied to problems with sufficient smoothness.

2.1. Gaussian Process (GP) Modeling

We begin by defining the input domain as $\Omega \subset \mathbb{R}^p$. Let $\mathbf{x}^{(i)}$ be samples from Ω , i.e. $\mathbf{x}^{(i)} \in \Omega \subset \mathbb{R}^p$. Assume that the data are given as $\mathcal{D}_{\text{train}} = (\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)}))$ for $i = 1, 2, \dots, N$, where $f(\mathbf{x}^{(i)})$ represents the value of the observed function at $\mathbf{x}^{(i)}$ and N is the number of (unique) samples. We denote the training dataset as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times p}, \quad \mathbf{f} = \begin{bmatrix} f(\mathbf{x}^{(1)}) \\ f(\mathbf{x}^{(2)}) \\ \vdots \\ f(\mathbf{x}^{(N)}) \end{bmatrix} \in \mathbb{R}^N.$$

The squared exponential (SE) kernel can be derived from the Reproducing Kernel Hilbert Space (RKHS), where a positive definite kernel defines an inner product between functions. The SE kernel can be defined as:

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}|^2}{2\delta^2}\right),$$

where δ is the kernel length scale that controls the smoothness of the function. In this work, δ is set by an optimizer that finds the best δ values within the range $[0.001, 1000]$ based on the various hyperparameters of our scheme. Based on our experiments, the optimizer found δ to be in the range $[1.0, 10.0]$. Using the kernel function $k(\cdot, \cdot)$ defined above, we can form the $(N \times N)$ covariance matrix, $\mathbf{K}_{[f,f]}$, computed using the training points, $\{\mathbf{x}^{(i)}\}_{i=1}^N$. If we assume that our approximation is of the form:

$$u(\mathbf{x}^*) = \sum_{i=1}^N \hat{u}_i \phi_i(\mathbf{x}^*),$$

where \hat{u}_i denotes the coefficients of our expansion and $\phi_i(\cdot)$ denotes our basis functions, then our coefficient vector is given by the expression $\hat{\mathbf{u}} = \mathbf{K}^{-1}\mathbf{f}$ and $\phi_i(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^{(i)})$, where \mathbf{K}^{-1} is the inverse of the covariance matrix. For noisy observations, the posterior variance at a test point \mathbf{x}^* is given by the following expression:

$$\text{Var}[u(\mathbf{x}^*)] = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{K}(\mathbf{x}^*, \mathbf{X})\mathbf{K}(\mathbf{X}, \mathbf{X})^{-1}\mathbf{K}(\mathbf{X}, \mathbf{x}^*),$$

where $\mathbf{K}(\mathbf{x}^*, \mathbf{X}) = [k(\mathbf{x}^*, \mathbf{x}^{(1)}), \dots, k(\mathbf{x}^*, \mathbf{x}^{(N)})]$. However, if the observations at the training points are noise-free, $\text{Var}[u(\mathbf{x}^*)] = 0$, which is equivalent to the interpolation using radial basis functions (RBFs).

A GP is a stochastic process defined as a collection of random variables indexed by time, space, or a more general input domain, such that any finite collection of these variables follows a joint multivariate Gaussian distribution. This property makes GPs particularly powerful for modeling unknown functions in a nonparametric Bayesian framework. GPs can be modeled by placing a prior over the latent function, \mathbf{f} , as:

$$f(x) \sim \mathcal{GP}(\mu(\cdot), \text{cov}(\cdot, \cdot)),$$

where $\mu(\cdot)$ is the mean function of the process and $\text{cov}(\cdot, \cdot)$ is its covariance function. In practice, one often sets $\mu(\cdot) = 0$ and uses the kernel function as the covariance function, i.e., $\text{cov}(f(\mathbf{x}^{(i)}), f(\mathbf{x}^{(j)})) = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$. As discussed in [14], under the GP prior, the function values at \mathbf{f} follow a multi-variate Gaussian distribution, $\mathcal{P}(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{0}, k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}))$. This is commonly referred to as GP projection. Let us assume that we want to compute the distribution of the function value at any input, \mathbf{x} , namely $\mathcal{P}(f(\mathbf{x}) | \mathbf{f})$. Since \mathbf{f} and $f(\mathbf{x})$ are both assumed to follow a multivariate Gaussian distribution, we obtain a conditional Gaussian:

$$\mathcal{P}(f(\mathbf{x}) | \mathbf{f}) = \mathcal{N}(f(\mathbf{x}) | \mu(\mathbf{x}), \sigma^2(\mathbf{x})),$$

where the conditional mean and variance, respectively, are given by

$$\mu(\mathbf{x}) = \text{cov}(f(\mathbf{x}), \mathbf{f}) \mathbf{K}^{-1} \mathbf{f},$$

and

$$\sigma^2(\mathbf{x}) = \text{cov}(f(\mathbf{x}), f(\mathbf{x})) - \text{cov}(f(\mathbf{x}), \mathbf{f}) \mathbf{K}^{-1} \text{cov}(\mathbf{f}, f(\mathbf{x})).$$

In the expression above, $\text{cov}(f(\mathbf{x}), \mathbf{f}) = k(\mathbf{x}, \mathbf{X}) = [k(\mathbf{x}, \mathbf{x}^{(1)}), \dots, k(\mathbf{x}, \mathbf{x}^{(N)})]$ and $\sigma(\cdot)$ denotes the standard deviation.

Since we are using a squared-exponential kernel in this work, the GP prior enforces smoothness and infinite differentiability on the latent function. In the case of traditional GPs, we assume noisy measurements represented as follows:

$$y_{(i)} = f(\mathbf{x}^{(i)}) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_{n,i}^2)$$

where each observation has its own (known or estimated) noise variance, $\sigma_{n,i}^2$, allowing for heteroscedastic noise.

We assume we are handling uncorrelated additive noise of the form:

$$\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}.$$

where

$$\mathbf{y} = \begin{bmatrix} y_{(1)} \\ y_{(2)} \\ \vdots \\ y_{(N)} \end{bmatrix} \in \mathbb{R}^N, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{(1)} \\ \epsilon_{(2)} \\ \vdots \\ \epsilon_{(N)} \end{bmatrix} \in \mathbb{R}^N.$$

Since both \mathbf{f} and $\boldsymbol{\epsilon}$ are assumed to be Gaussian, \mathbf{y} is also Gaussian. Thus, \mathbf{y} can be denoted as:

$$\mathbf{y} \sim \mathcal{N}(0, \mathbf{K}_{ff} + \sigma_{n,i}^2 \mathbf{I})$$

where \mathbf{K}_{ff} is the GP covariance matrix computed using the squared-exponential kernel and \mathbf{I} is the identity matrix. This expression can be further simplified to the following:

$$\mathbf{y} \sim \mathcal{N}(0, \mathbf{K}_{ff} + \mathbf{R})$$

where, $\mathbf{R} = \sigma_{n,i}^2 \mathbf{I}$, is the diagonal heteroscedastic noise matrix.

The mean of the GP posterior acts as the approximator of the latent function. In our case of GP with independent heteroscedastic noise, the posterior mean can be expressed as follows:

$$\bar{f}(\mathbf{x}) = \mathbf{K}(\mathbf{x}^*, \mathbf{X})[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \mathbf{R}]^{-1} \mathbf{y}, \quad (1)$$

where $\mathbf{K}(\mathbf{x}^*, \mathbf{X})$ denotes the covariance vector between the test point \mathbf{x}^* and the training data \mathbf{X} . Note that upon comparison with the noise-free expression previously given, the matrix $\mathbf{K}((\mathbf{X}, \mathbf{X}) + \mathbf{R})$ contains the additional \mathbf{R} term used to model the presence of noise (an observation relevant to our inversion and sparsification discussion below).

2.2. Derivative-Enhanced Gaussian Process (GP) Surrogate Modeling for Noisy Data

Assuming that we have access to derivative information of \mathbf{f} up to an arbitrary order d , the accuracy of the GP can be improved by incorporating these derivatives into the training procedure [15, 16, 17, 18, 19, 20]. It is important to note that the kernel function used to model the GP should be sufficiently smooth and differentiable, which is consistent with our choice of the squared-exponential kernel. An alternative choice found in the literature is the family of Matérn kernels [21].

A GP that leverages derivative information can be formulated as $\mathbf{F} \sim \mathcal{GP}(0, \mathbf{K}_{[f, \nabla f, \nabla^2 f, \dots, \nabla^d f]} + \mathbf{R})$, where \mathbf{F} is a vector that contains noisy observations of \mathbf{f} and its derivatives, $\mathbf{K}_{[f, \nabla f, \nabla^2 f, \dots, \nabla^d f]}$ is the covariance matrix with derivatives, and \mathbf{R} is the diagonal matrix containing the noise variances of each observed quantity. The formulation of \mathcal{GP} is structured as follows:

$$\begin{bmatrix} f \\ \nabla f \\ \nabla^2 f \\ \vdots \\ \nabla^d f \end{bmatrix} \sim \mathcal{GP} \left(0, \begin{bmatrix} \mathbf{K}_{[f,f]} + \sigma_f^2 \mathbf{I} & \mathbf{K}_{[f,\nabla f]} & \mathbf{K}_{[f,\nabla^2 f]} & \cdots & \mathbf{K}_{[f,\nabla^d f]} \\ \mathbf{K}_{[\nabla f,f]} & \mathbf{K}_{[\nabla f,\nabla f]} + \sigma_{\nabla f}^2 \mathbf{I} & \mathbf{K}_{[\nabla f,\nabla^2 f]} & \cdots & \mathbf{K}_{[\nabla f,\nabla^d f]} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}_{[\nabla^d f,f]} & \mathbf{K}_{[\nabla^d f,\nabla f]} & \mathbf{K}_{[\nabla^d f,\nabla^2 f]} & \cdots & \mathbf{K}_{[\nabla^d f,\nabla^d f]} + \sigma_{\nabla^d f}^2 \mathbf{I} \end{bmatrix} \right) \quad (2)$$

where $\nabla^d f$ represents the d^{th} derivative of function \mathbf{f} . The matrix $\mathbf{K}_{[\nabla^n f, \nabla^m f]}$ corresponds to the covariance of the n^{th} and m^{th} derivative observations; the elements are calculated using the derivatives of the kernel. Details regarding derivatives of the squared-exponential kernel can be found in [15, 16, 20]. For $d > 1$, let $\nabla^d f$ denote a column vector containing (unique) derivative terms. For example, in our notation, $\nabla^2 f$ of a 2D function is written as:

$$\nabla^2 f = \left[\frac{\partial^2 f}{\partial x_1^2}, \frac{\partial^2 f}{\partial x_1 \partial x_2}, \frac{\partial^2 f}{\partial x_2^2} \right]^T. \quad (3)$$

The covariance matrix presented in Eq. 2 can be interpreted as an exact GP covariance matrix perturbed by a diagonal regularizer (e.g., *nugget*) $\sigma_f^2 \mathbf{I}, \sigma_{\nabla f}^2 \mathbf{I}, \dots, \sigma_{\nabla^d f}^2 \mathbf{I}$. In the limiting case and given sufficient smoothness, $\lim_{\sigma_f^2, \sigma_{\nabla f}^2, \dots, \sigma_{\nabla^d f}^2 \rightarrow 0} \text{GP}$, we recover the exact interpolation of the noise-free GP, i.e.,

$$\text{GP}_{\text{noisy}} \xrightarrow{\sigma_f^2, \sigma_{\nabla f}^2, \dots, \sigma_{\nabla^d f}^2 \rightarrow 0} \text{GP}_{\text{noise-free}}.$$

From a computational perspective, the additive noise terms, $\sigma_f^2 \mathbf{I}, \sigma_{\nabla f}^2 \mathbf{I}, \dots, \sigma_{\nabla^d f}^2 \mathbf{I}$, act similarly to Tikhonov regularization in kernel ridge regression [22] where the addition of derivative noise introduces a form of regularization, and correspondingly uncertainties in higher-order derivatives impose smoothness constraints on the posterior mean. This point will be relevant to interpreting some of our results in Section 5.

To extend the formulation in Eq. 1 to the derivative-informed setting, we modify \mathbf{X} and \mathbf{y} in Eq. 1 to include the derivatives as follows:

$$\mathbf{X}_{\text{der}} = \begin{bmatrix} \mathbf{X}^{(0)} \\ \mathbf{X}^{(1)} \\ \vdots \\ \mathbf{X}^{(d)} \end{bmatrix}, \quad \mathbf{y}_{\text{der}} = \begin{bmatrix} f(\mathbf{X}^{(0)}) \\ \nabla f(\mathbf{X}^{(1)}) \\ \vdots \\ \nabla^d f(\mathbf{X}^{(d)}) \end{bmatrix},$$

where $\mathbf{X}^{(k)}$ denotes the set of input points at which the derivatives of order d are presented. Using this notation, the posterior mean for the derivative-informed GP setting can then be expressed as:

$$\tilde{f}(\mathbf{x}) = \mathbf{K}_{\text{der}}(\mathbf{x}^*, \mathbf{X}_{\text{der}}) [\mathbf{K}_{\text{der}}(\mathbf{X}_{\text{der}}, \mathbf{X}_{\text{der}}) + \mathbf{R}]^{-1} \mathbf{y}_{\text{der}},$$

where \mathbf{K}_{der} is the covariance matrix from Eq. 2 and $\mathbf{K}_{\text{der}}(\mathbf{x}^*, \mathbf{X}_{\text{der}})$ denotes the derivative-informed covariance vector between the test point, \mathbf{x}^* , and \mathbf{X}_{der} .

This formulation, which includes derivative information, preserves the interpolatory structure of the derivative-free GP model, i.e., the posterior mean $\tilde{f}(\mathbf{x})$ remains a kernel-based interpolant in the noise-free setting, and the inclusion of \mathbf{R} maintains regularization. By incorporating derivative information through \mathbf{K}_{der} , the mean function becomes a ‘‘curvature-aware interpolant’’.

2.3. Noise-Free Derivative-Enhanced Gaussian Process (GP) Surrogate Modeling

When there is no noise present in the data, i.e., when the noise variance terms in Eq. 2 are set to zero, Eq. 2 can be modified for the noise-free case as:

$$\begin{bmatrix} f \\ \nabla f \\ \nabla^2 f \\ \vdots \\ \nabla^d f \end{bmatrix} \sim \mathcal{GP} \left(0, \begin{bmatrix} \mathbf{K}_{[f,f]} & \mathbf{K}_{[f,\nabla f]} & \mathbf{K}_{[f,\nabla^2 f]} & \cdots & \mathbf{K}_{[f,\nabla^d f]} \\ \mathbf{K}_{[\nabla f,f]} & \mathbf{K}_{[\nabla f,\nabla f]} & \mathbf{K}_{[\nabla f,\nabla^2 f]} & \cdots & \mathbf{K}_{[\nabla f,\nabla^d f]} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}_{[\nabla^d f,f]} & \mathbf{K}_{[\nabla^d f,\nabla f]} & \mathbf{K}_{[\nabla^d f,\nabla^2 f]} & \cdots & \mathbf{K}_{[\nabla^d f,\nabla^d f]} \end{bmatrix} \right). \quad (4)$$

The covariance matrix mentioned in Eq. 4 remains symmetric and positive definite, as guaranteed by the following lemma.

LEMMA 2.3.1. [Positive definiteness of the derivative-informed covariance matrix] Let \mathbf{K}_{der} be the covariance matrix of a GP constructed from the function f and the function f derivatives

$\nabla f, \dots, \nabla^d f$ up to order d , using a smooth, positive-definite kernel $k(\cdot, \cdot)$. Then \mathbf{K}_{der} is symmetric and positive definite.

Proof. See Appendix C.1. □

Including higher-order derivatives is shown to reduce the prediction error, as described in the following lemma.

LEMMA 2.3.2. *[Error estimate of derivative-informed GP] Let f denote the true function observation and \hat{f}^d be the GP prediction using derivatives up to order d . Then, the mean squared error (MSE) satisfies the following inequality:*

$$MSE(\hat{f}^d) \leq MSE(\hat{f}^{d-1}),$$

i.e., including higher-order derivatives reduces or maintains the MSE.

Proof. See Appendix C.2. □

Including derivatives is shown to preserve the exponential decay of the squared-exponential (SE) kernel, ensuring that all covariance entries and their derivative blocks decay exponentially with the inter-point distance, as shown by the following lemma.

LEMMA 2.3.3. *[Exponential decay of the squared-exponential kernel derivatives] Let $k : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}$ be the squared-exponential (SE) kernel defined as*

$$k(\mathbf{x}, \mathbf{y}) = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\delta^2}\right),$$

and let d be the order of derivatives, then for any multi-indices α, β with $|\alpha|, |\beta| \leq d$, there exist constants $C_{\alpha, \beta} > 0$ and $\gamma = \frac{1}{2\delta^2} > 0$ such that for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$

$$|\partial_{\mathbf{x}}^{\alpha} \partial_{\mathbf{y}}^{\beta} k(\mathbf{x}, \mathbf{y})| \leq C_{\alpha, \beta} \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2).$$

Therefore, covariance entries and their derivative blocks decay exponentially with the squared inter-point distance.

Proof. See Appendix C.3. □

In this work, the maximum number of derivatives is set to four, and the squared-exponential kernel was used to calculate the elements of the $\mathbf{K}_{[f, \nabla f, \nabla^2 f, \dots, \nabla^d f]}$ matrix.

Under the noise-free regression assumption – i.e., we observe the true function values without additive measurement noise – the GP prior implies a joint Gaussian distribution over the training outputs \mathbf{f} and the test outputs \mathbf{f}^* at M test points (\mathbf{X}^*):

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} = \mathcal{GP}\left(0, \begin{bmatrix} \mathbf{K}_{[f, f]} & \mathbf{K}_{[f, f^*]} \\ \mathbf{K}_{[f^*, f]} & \mathbf{K}_{[f^*, f^*]} \end{bmatrix}\right), \quad (5)$$

where $\mathbf{K}_{[f, f]}$ is the covariance matrix ($N \times N$) computed between the training points, $\mathbf{K}_{[f^*, f^*]}$ is the covariance matrix ($M \times M$) computed between the testing points and $\mathbf{K}_{[f, f^*]}$ (and its transpose, $\mathbf{K}_{[f^*, f]}$) are the cross-covariance matrices ($N \times M$) and ($M \times N$) respectively between the training and testing points.

Conditioning on the observed training data yields the predictive posterior for the latent function at the test inputs:

$$(\mathbf{f}^* \mid \mathbf{X}, \mathbf{f}, \mathbf{X}^*) \sim \mathcal{N}(\bar{\mathbf{f}}^*, \text{cov}(\mathbf{f}^*)),$$

with

$$\bar{\mathbf{f}}^* = \mathbf{K}_{f^*,f} \mathbf{K}_{f,f}^{-1} \mathbf{f}, \quad \text{cov}(\mathbf{f}^*) = \mathbf{K}_{f^*,f^*} - \mathbf{K}_{f^*,f} \mathbf{K}_{f,f}^{-1} \mathbf{K}_{f,f^*}.$$

Since we are in a noise-free setting, there is no observation noise variance, $\sigma_{n,i}^2$, added to the diagonal of $\mathbf{K}_{[f,f]}$, and the predictive mean $\bar{\mathbf{f}}^*$ exactly interpolates the training data at any training input ($\mathbf{x}^{(i)}$), ($\bar{\mathbf{f}}^*(\mathbf{x}^{(i)}) = \mathbf{f}(\mathbf{x}^{(i)})$). The predictive covariance collapses to zero at the training points, reflecting the certainty about the true function values at those locations.

This formulation makes GPs particularly appealing for deterministic function approximation (e.g., interpolating solutions of differential equations, modeling smooth physical phenomena without measurement noise), where the GP serves as a nonparametric interpolator with built-in uncertainty quantification away from the training data.

2.4. Numerical Verification Experiments

GPs augmented with derivative measurements up to 4th order are verified on four numerical functions: 1D, 2D, 3D Griewank functions, and 3D Rosenbrock function. The primary reason for choosing the above functions is that they are challenging from an interpolation perspective while still maintaining smooth derivatives. Additionally, the dimensionality of the Griewank function can be increased without a significant change in the function form, which allows us to understand how the GP with derivatives scales with dimensionality. For all these functions, experiments are conducted by increasing the number of training points and the order of derivative information at each training point. The training dataset, \mathcal{D}_{train} , is generated by selecting equi-spaced points within the domain. The range of the input features is set at $[-\pi, \pi]$ for the Griewank functions (1D, 2D, and 3D) and $[-5, 10]$ for the Rosenbrock function. Using the sampled \mathbf{X}_{train} and \mathbf{f}_{train} , we compute the derivatives of \mathbf{f}_{train} with respect to the input features up to the 4th order analytically. The analytical expressions for the derivatives of the Griewank function can be found in [23]. The trained GP is tested on a dataset that includes 1000 randomly generated points, $\mathcal{D}_{test} = \{\mathbf{X}_{test}, \mathbf{f}_{test}\}$, within the trained domain. During training, the kernel length scale, δ , is optimized for lower prediction error on the test dataset. We note that \mathbf{f}_{test} does not include any derivatives, and all the prediction errors are reported in mean squared error (MSE). The primary reason for reporting prediction errors in MSE rather than RMSE is that MSE shows differences in error magnitudes clearly, especially at smaller scales (e.g., for the order of 10^{-19}). Since the goal of the plots is to compare the decrease in prediction error with the number of training points and derivative orders, the MSE provides a clearer separation of trends on the y-axis. In contrast, RMSE would compress these differences by taking a square root, making the patterns less visible.

Results of the experiments are shown in Fig.2. For all the studied functions, the prediction error reduced when the number of training points and the order of derivatives are increased. For the 1D Griewank function with three points, the prediction error reduced from approximately 10^{-3} , without derivatives, to 10^{-15} when trained with 4th order derivatives. As the number of training points increased, the prediction error reduced for GP with and without derivatives. However, the prediction error starts to plateau as the number of training points increases beyond six for the

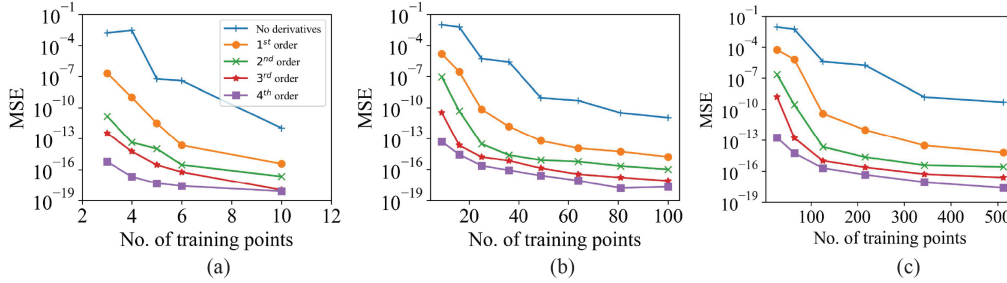


Figure 2: Results of numerical experiments for varying number of training points and derivative order (a) 1D Griewank function, (b) 2D Griewank function, and (c) 3D Griewank function.

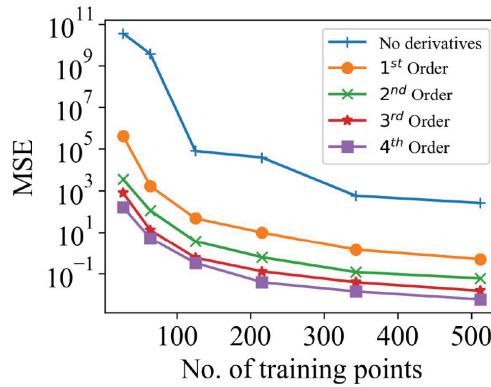


Figure 3: Results of numerical experiments on 3D Rosenbrock function for varying number of training points and order of derivatives. Note that the order of MSE errors is larger due to the steeper and higher magnitude nature of the Rosenbrock function. Although the magnitude of MSE errors is larger, the GP model performs equally well in relative terms when compared with the predictive performance of the Griewank function in Fig.2.

models trained with derivatives. This is due to the fact that the prediction error is already in the range of 10^{-17} , and further increase in the number of training points would not improve the accuracy further. Similar observations can be noticed for 2D and 3D Griewank functions. For the 3D Griewank function trained with 27 points, increasing order of derivatives reduced the prediction error from 10^{-2} (for no derivatives) to 10^{-13} (for 4th order). For the Rosenbrock function with 27 training points, including derivative information reduced the prediction error from 10^{10} to less than 10^2 when 4th order derivatives are included in training. A similar observation can be made for higher N values.

Furthermore, to investigate the effect of adding noise to the GPs augmented with derivative measurements on numerical stability and predictive performance, we perform additional experiments. We examine the behavior of the matrix condition number (in L_2) of the noisy covariance matrix, $\Sigma = \mathbf{K} + \mathbf{R}$, under varying nugget parameters, σ^2 .

Figure 4 shows the results of the numerical experiments for the effect of varying noise levels on the condition numbers. In the case of increasing noise, an incremental 1% increase in noise levels for each derivative order is added from 1% noise for 0th order derivative to 5% noise for 4th order derivative. Conversely, an incremental 1% decrease in noise is added from 5% noise

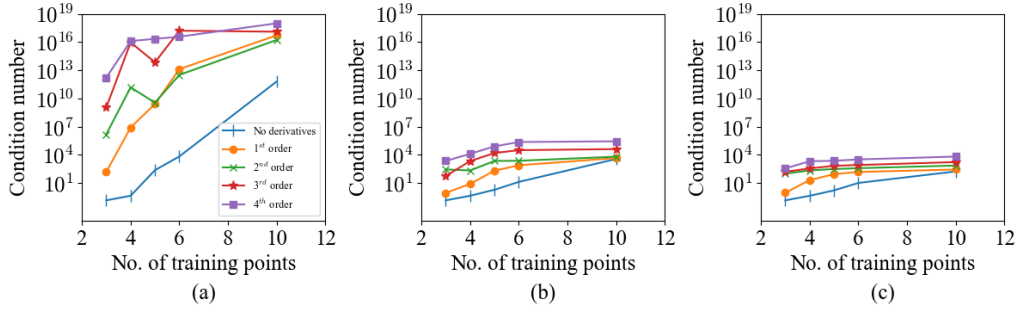


Figure 4: Results of numerical experiments for matrix conditioning on 1D Griewank function for varying number of training points, order of derivatives, and varying noise. (a) No noise, (b) Increasing noise, and (c) Decreasing noise.

for 4^{th} order derivative to 1% noise for 0^{th} order derivative. In the case of decreasing noise, the covariance matrices exhibit small condition numbers, leading to higher predictive errors as shown in Figure 5. In contrast, for increasing noise, the covariance matrices exhibit large condition numbers, resulting in lower predictive errors. This happens because the nugget and the scaling of the derivatives dominate the predictive error. In decreasing noise, the small nugget results in insufficient regularization of higher-order derivatives. However, in the case of increasing noise, the larger nugget for higher-order derivatives stabilizes the covariance matrices, which reduces the predictive error despite a higher condition number. The MSE begins to plateau in Figure 5 because the statistical component of noise dominates the overall error, and further regularization does not yield any improvement in accuracy.

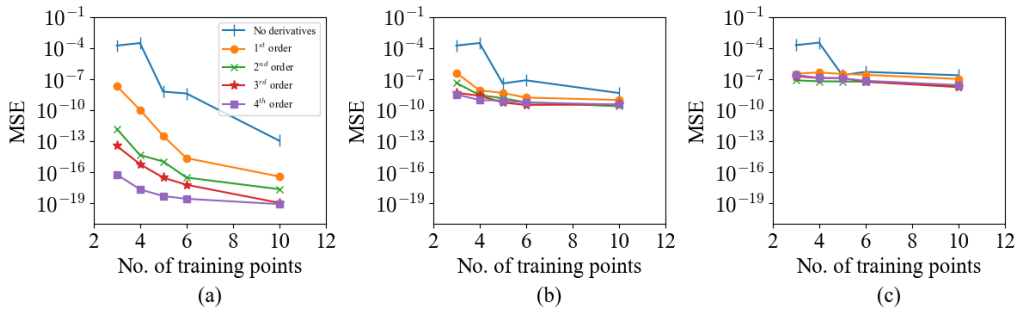


Figure 5: Results of numerical experiments for the predictive performance (measured by MSE) on 1D Griewank function for varying number of training points, order of derivatives, and varying noise. (a) No noise, (b) Increasing noise, and (c) Decreasing noise.

3. Sparse Cholesky for Derivative-Enhanced GP

In Section 2, we presented both traditional GP modeling and our enhancement using derivative information. As previously discussed, the incorporation of derivatives improves the accuracy of our GP approximation, given sufficient smoothness of the underlying function we are approximating. However, derivative-enhanced GPs come at a cost: the inclusion of derivative terms

leads both to a significant increase in the size of the covariance matrix and an increase in its conditioning as a function of the number of points N , dimension p , and derivative order d . Focusing on the former, the size of the system: the number of derivative terms, N_d , to be incorporated in the covariance matrix is given by the following lemma.

LEMMA 3.0.1. [Covariance matrix size and sparsity] Given N training points drawn from a p -dimensional space and with derivatives up to the order d , the number of covariance terms is given by the following:

$$N_d = \binom{p+d-1}{d} N, \quad (6)$$

where the resulting (updated) covariance matrix $\mathbf{K}_{der} \in \mathbb{R}^{N_d \times N_d}$ is symmetric positive-definite (s.p.d.) with a block structure corresponding to the derivatives.

Proof. See Appendix C.4. □

The conditioning of the updated system is given by the following:

LEMMA 3.0.2. [Conditioning of derivative-informed covariance matrix] For a derivative-informed covariance matrix \mathbf{K}_{der} with order d derivatives and kernel length scale l , the condition number is given by

$$\kappa(\mathbf{K}_{der}) = \|\mathbf{K}_{der}\|_2 \cdot \|\mathbf{K}_{der}^{-1}\|_2,$$

which increases with d and decreases with increasing l . Given that the covariance matrix is s.p.d., this expression can be rewritten as:

$$\kappa(\mathbf{K}_{der}) = \frac{\lambda_{max}}{\lambda_{min}},$$

where λ_{max} and λ_{min} denote the maximum and minimum eigenvalues, respectively, of the covariance matrix.

Proof. See Appendix C.5. □

Assuming all the derivative terms are available, computation of \mathbf{K}_{der}^{-1} while solving the exact GP scales as $O((N \times N_d)^3)$ in time and as $O((N \times N_d)^2)$ in memory. (We use ‘exact’ to denote the covariance matrix prior to any sparsification approximations as presented below). Clearly, efficient computation of \mathbf{K}_{der}^{-1} is desired. As discussed earlier, the size of the covariance matrix with derivatives increases significantly with respect to N , d , and p . Solving for such large matrix systems can be computationally intractable [24], and hence there is a need to efficiently approximate the covariance matrix to help alleviate these scalability issues. Solving these large matrix systems can be approximated through several methods [25], such as low rank approximation methods [26, 27], sparse approximation [28, 29, 30, 31, 32, 33], and others [34, 35]. Out of the several studied methods, work related to the Vecchia approximation has gained popularity in the world of GPs applied to computational and data science problems [28, 36, 30, 37, 38, 39]. Within this body of work, one of the critical aspects upon which these approximation methods are built is the choice of the ordering and conditioning set [40, 41, 42, 43]. In this work, we utilize sparse Cholesky factorization of the exact GP [44, 45], which has proven to be an efficient approximation method with near-linear computational complexity. One contribution of our work is to extend the aforementioned sparse Cholesky factorization approach to include derivative observations of arbitrary order d . In this section, we provide a brief overview of the sparse Cholesky factorization algorithm [44] and how we extend it to include derivative observations. We then

show verification results using our updated approach – highlighting the impact of, and interplay between, the various parameters of the algorithm such as the sparsification factor, number of training points, etc.

3.1. Sparse GP using Cholesky Factorization

The objective of the sparse Cholesky algorithm is to build a sparse precision matrix. The precision matrix, in the context of computational methods, refers to the inverse of the covariance matrix obtained as part of our GP model. In this section, we provide a brief review of the steps involved in building a sparse GP. For additional details, readers are directed to the original work upon which our work is based [44, 45].

The first step in building a sparse precision matrix involves ordering the set of training points, $\{\mathbf{x}^{(i)}, i \in I\}_{i=1}^N$, using maximum-minimum distance (MMD) ordering [44, 45, 41]. In MMD ordering, the sequence of the points is chosen based on the maximum of the minimum distances from the set of unordered points, and the index of the ordered points is stored as a vector \mathbf{P} , as shown below:

$$\mathbf{P}(q+1) = \arg \max_{i \in I \setminus \{1, \dots, q\}} \text{dist}(\mathbf{x}^{(i)}, \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(q)}\}) \quad (7)$$

The length scale of the ordered points is stored in a vector \mathbf{l} , given as:

$$\mathbf{l}^{(i)} = \text{dist}(\mathbf{x}^{\mathbf{P}(i)}, \{\mathbf{x}^{\mathbf{P}(1)}, \dots, \mathbf{x}^{\mathbf{P}(i-1)}\}). \quad (8)$$

The intuition behind this step is that the ordering is performed by selecting the points furthest from the previous point; thus, the reordered sequence contains points that are “not too close to each other.” The aforementioned ordering has been shown to produce Cholesky factors with near sparsity, the proof of which can be found in [44, 45].

Once the ordering is determined, the sparsity set (S) is determined by:

$$S_{\mathbf{P}, \mathbf{l}, \rho} = \{(i, j) \subset I \times I : i \leq j, \text{dist}(\mathbf{x}^{\mathbf{P}(i)}, \mathbf{x}^{\mathbf{P}(j)}) \leq \rho \mathbf{l}^{(j)}\}, \quad (9)$$

where ρ influences the size of the sparsity set. The sparsity set we obtain can be aggregated into groups based on both the ordering and geometric location, denoted by $S_{\mathbf{P}, \mathbf{l}, \rho, \lambda}$, where λ is set as 1.5 as suggested by [44]. These aggregated groups are termed supernodes, \mathcal{SN} . Each \mathcal{SN} consists of a list of parent and child indices, where the term parent refers to the index of the columns in the matrix and the term child denotes the indices of non-zero entries in the column. The concept of supernodes is particularly useful in this work, as they allow us to reuse the computed Cholesky factors for a set of rows and columns within the matrix. This offers a significant computational advantage in our digital twin framework, which will be discussed in detail later.

Based on the determined ordering and sparsity pattern (with aggregation), the sparse matrix is obtained by KL minimization, given by the following expression:

$$\mathbf{U} = \arg \min_{\hat{\mathbf{U}} \in S_{\mathbf{P}, \mathbf{l}, \rho, \lambda}} \mathcal{D}_{KL} \left(\mathcal{N}(0, \mathbf{K}) \parallel \mathcal{N}(0, (\hat{\mathbf{U}} \hat{\mathbf{U}}^T)^{-1}) \right). \quad (10)$$

The above equation has a closed-form solution which can be found in [44, 45]. In the case where there is noise in the data, the noise can be represented by a diagonal heteroscedastic noise matrix, \mathbf{R} , where the diagonal elements of \mathbf{R} are given by $\sigma_{n,i}^2 \mathbf{I}$. As discussed in [44], \mathbf{R} attenuates the exponential decay of the matrix $\Sigma = \mathbf{K} + \mathbf{R}$, where \mathbf{K} is the original covariance matrix of the GP without any derivative information. Thus, Σ has been decomposed into $\Sigma \approx (\mathbf{L}\mathbf{L}^T)^{-1} \hat{\mathbf{L}} \hat{\mathbf{L}}^T \mathbf{R}$, and an incomplete Cholesky factorization with zero fill-in has been applied in [44]. The details

of the incomplete Cholesky factorization with zero fill-in are mentioned in [44, 46]. The proof of the decomposition of Σ is mentioned in Appendix B.2. In the case when there is no noise in the data, the \mathbf{R} matrix becomes a zero matrix, in which case, $\Sigma = \mathbf{K}$. Then Σ decays exponentially without the need to be decomposed into $\Sigma \approx (\mathbf{L}\mathbf{L}^\top)^{-1}\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top\mathbf{R}$.

In this work, we follow the steps mentioned in this section to generate a sparse approximation, \mathbf{U} , of the precision matrix. The algorithm is implemented in Python (a GitHub link to the codebase will be made available after reviews are completed and the paper is accepted for publication), and the code results are compared with the original work’s results by [45]. Although we primarily focus on noise-free data in this work, we also present a discussion of noisy data with derivative information.

3.2. Derivative-Enhanced Sparse GP for Noise-Free Data

In this section, the sparse GP algorithm presented in Section 3.1 is extended to include derivative measurements up to an arbitrary derivative order, d . As discussed in Section 2, incorporating derivatives does not violate the symmetric positive-definite property of the covariance matrix; thus, extending the idea of sparse GP with Cholesky factorization to incorporate derivatives is valid. A critical point to be addressed when incorporating the derivative observations into the sparse GP approximation is how the derivatives are incorporated into the ordering, \mathbf{P} , and its corresponding sparsity set. In an exact GP scenario, the way derivatives are placed in the matrix formation does not have any effect on the mathematical accuracy, as matrix ordering does not change the spectrum of the operator; however, that is not the case for sparse GP. The following sections address how derivatives are included in building the sparse GP. The DT application of derivative-enhanced sparse GP for noise-free data is detailed in Section 5.

3.2.1. Ordering Derivatives and Supernodes with Derivatives

Given a set of functional and derivative measurements, \mathbf{F} , up to an arbitrary derivative order, d , at all training points, we studied four different methods of ordering with derivatives. In this section, we discuss only one method in detail, which we call “point-wise ordering algorithm 1”. The readers are referred to Appendix A.1 for the rest of the three methods, namely “point-wise ordering algorithm 2”, “measurement-wise ordering algorithm 1”, and “measurement-wise ordering algorithm 2”. In Appendix A.1, we provide a detailed comparison between all four methods.

In “point-wise ordering algorithm 1”, the derivative measurements are grouped with the points in an array-of-structures format. For better understanding, we illustrate the structure of the covariance matrix that has the function values $f(x)$ and its corresponding first-order derivative measurement $\nabla f(x)$ in Fig. 6. Assuming $N=10$, the plot shows the structure of the \mathbf{K}_{der} matrix when f and ∇f are grouped by points. Here, \mathbf{F} is ordered as, $[f^{(\mathbf{P}(1))}, \nabla f^{(\mathbf{P}(1))}, f^{(\mathbf{P}(2))}, \nabla f^{(\mathbf{P}(2))}, \dots, f^{(\mathbf{P}(N))}, \nabla f^{(\mathbf{P}(N))}]$.

The initial ordering \mathbf{P} was obtained using Eq. 7 without any derivative measurements, and then we extend \mathbf{P} to incorporate derivative measurements to obtain \mathbf{P}^d . A subscript $po-1$ is added to \mathbf{P}^d to refer to the ordering grouped by the point-wise ordering algorithm 1. The algorithm used to obtain \mathbf{P}_{po-1}^d is shown in Algorithm 1. The \mathbf{P}_{po-1}^d is obtained by iterating through \mathbf{P} and adding the index of each derivative measurement immediately after the point measurement in the sequence.

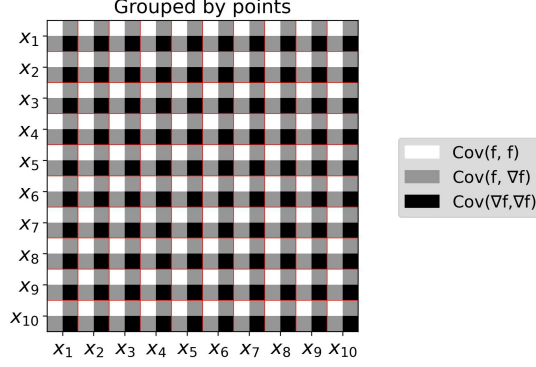


Figure 6: The figure illustrates the point-wise ordering algorithm 1 of incorporating derivative measurements in the formation of the \mathbf{K}_{der} matrix. The derivative measurements are grouped by points as they are placed next to the functional observations. Note that these functional observations are ordered according to \mathbf{P} .

Algorithm 1 Constructing the \mathbf{P}_{po-1}^d array

- 1: **Input:** \mathbf{P} from MMD ordering
 - 2: **Output:** \mathbf{P}_{po-1}^d
 - 3: $td \leftarrow \lfloor N_d/N \rfloor$
 - 4: **for** $i \leftarrow 1$ to N **do**
 - 5: **for** $j \leftarrow 1$ to td **do**
 - 6: $k \leftarrow i \cdot td + j$
 - 7: $\mathbf{P}_{po-1}^d[k] \leftarrow \mathbf{P}[i] + N \cdot j$
 - 8: **end for**
 - 9: **end for**
-

The supernodes, \mathcal{SN} , are originally obtained without any derivative measurement, through the procedure described in Section 3.1. The existing supernodes, \mathcal{SN} , are then updated to include the derivative measurements to obtain \mathcal{SN}_{po-1}^d . In \mathcal{SN} , the set of parents and children consists of the index of the elements in \mathbf{P} . To obtain \mathcal{SN}_{po-1}^d , indices in each set of both parent and child are expanded to include the derivative measurements of the corresponding indices. Note that \mathcal{SN}_{po-1}^d is a list of multiple supernodes that are used to build the sparse matrix.

3.3. Derivative-Enhanced Sparse GP for Noisy Data

Following [44], when there is heteroscedastic noise, we can represent uncorrelated noise as a diagonal matrix, \mathbf{R} , where the diagonal elements of \mathbf{R} are given by $\sigma_{n,i}^2 \mathbf{I}$. Then, the noisy covariance matrix is given by $\mathbf{\Sigma} = \mathbf{K} + \mathbf{R}$, which can be decomposed into $\mathbf{\Sigma} \approx (\mathbf{L}\mathbf{L}^\top)^{-1} \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top \mathbf{R}$, and can algorithmically be solved using an incomplete Cholesky factorization with zero fill-in mentioned in [44]. The proof of the decomposition of $\mathbf{\Sigma}$ is mentioned in Appendix B.2. In the proof, we made the approximation that $\mathbf{K}\mathbf{R} \approx \mathbf{R}\mathbf{K}$ to reach the decomposition of $\mathbf{\Sigma}$, which led us to implement the incomplete Cholesky factorization mentioned in [44]. In the case when the covariance matrix is augmented to include derivative terms, as mentioned in Eq.4, and

the noise matrix, \mathbf{R} , is a block diagonal matrix, the approximation $\mathbf{K}_{der}\mathbf{R} \approx \mathbf{R}\mathbf{K}_{der}$ does not hold true, where \mathbf{K}_{der} is the derivative augmented covariance matrix. Since the approximation, $\mathbf{K}_{der}\mathbf{R} \approx \mathbf{R}\mathbf{K}_{der}$ does not hold true, $\Sigma_{der} \neq (\mathbf{L}\mathbf{L}^\top)^{-1}\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top\mathbf{R}$, where $\Sigma_{der} = \mathbf{K}_{der} + \mathbf{R}$.

To mitigate this issue, we decompose Σ_{der} as $\Sigma_{der} \approx (\mathbf{L}'\mathbf{L}'^\top)^{-1}$, where $\mathbf{L}' = \mathbf{R}^{-1/2}\mathbf{L}''$ and $(\mathbf{L}''\mathbf{L}''^\top) \approx (\mathbf{R}^{-1/2}\mathbf{K}_{der}\mathbf{R}^{-1/2} + \mathbf{I})^{-1}$. Using this decomposition, Σ_{der} can algorithmically be solved using the incomplete Cholesky factorization with zero fill-in. The readers are referred to Appendix B.3 for the details of the decomposition of Σ_{der} . For the purpose of this work, we primarily focus on the case of noise-free data. However, we present numerical experiments for the case of noisy data in Section 3.5.

3.4. Numerical Verification Experiments: Derivative-Enhanced Sparse GP with Noise-Free Data

We verify our derivative-enhanced sparse GP algorithm on the 1D, 2D, and 3D Griewank functions for varying numbers of training points and orders of derivatives. We assume no noise is present in the training data (either in the function values or their derivatives). Additionally, we also study the accuracy of the sparse GP with different ρ values to understand the influence of the sparsity of the matrix on the prediction accuracy.

We present in Figure 7 the results of our experiments for the Griewank functions with the sparsification factor set to $\rho = 10$. The training and testing data are the same as those used for training the exact GP reported in Section 2. For the reported hyperparameters, prediction from the sparse GP is equal to the exact GP (Fig. 2), and the observations reported from the results of the exact GP apply to the sparse GP as well. As the number of training points and the order of the derivative increase, the prediction error reduces significantly. Note that the grouping method has no appreciable effect on the prediction error, as the sparsity of the matrix is relatively small for the reported hyperparameters. Additionally, the higher dimensionality of the input function leads to a slightly higher predictive error, as shown in Figure 7.

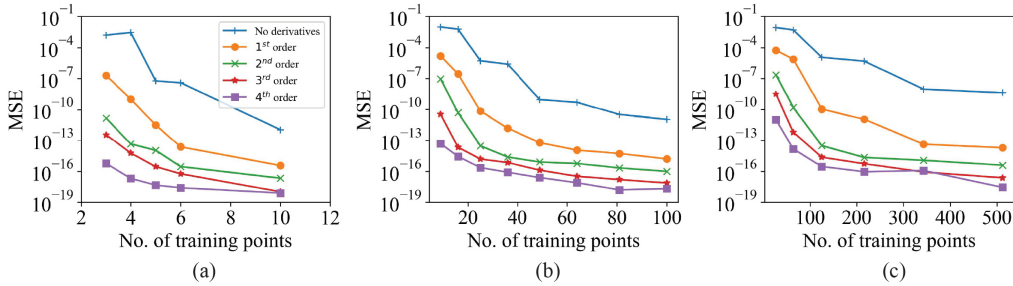


Figure 7: Prediction error from sparse GP for 1D (a), 2D (b), and 3D (c) Griewank functions

We know that the ordering of the derivatives within the matrix plays no role in the formal accuracy of the method due to spectral equivalency under permutations; however, this property does not hold when the matrix is made sparse. We perform additional experiments by varying ρ values to compare the predictive performance for the point-wise ordering algorithm 1 and the measurement-wise ordering algorithm 1. Similarly, we perform experiments by varying the number of training points, N , to compare the predictive performance of the point-wise ordering algorithm 1 and the measurement-wise ordering algorithm 1. These additional results are reported in Appendix A.2.

From the experiments given in Appendix A.1, it is evident that the point-wise ordering algorithm 1 and the measurement-wise ordering algorithm 2 show better predictive performance compared to the other proposed methods. However, in the case of sparse GP for our DT application, point-wise ordering algorithm 1 offers a computational advantage, as it allows adding additional functional and derivative observations by including additional columns in the matrix. Thus, point-wise ordering algorithm 1 is chosen as the main method in this work. The details of our dynamic algorithm are presented in Section 4.

3.5. Numerical Experiments: Derivative-Enhanced Sparse GP with Noisy Data

We perform experiments to investigate the predictive performance of adding increasing noise to the sparse GPs augmented with derivative measurements using the mathematical framework described in Section 3.3. An incremental 1% increase in noise levels for each derivative order is added from 1% noise for 0th order derivative to 5% noise for 4th order derivative while keeping $\rho = 10$. The results shown in the Fig.8 demonstrate that the mathematical framework described in Section 3.3 decreases the predictive error with increasing number of training points and increasing order of derivative measurements. A detailed comparison using numerical experiments between our method mentioned in Section 3.3 and the incomplete Cholesky factorization with zero fill-in described in [44, 46] has been left for future work.

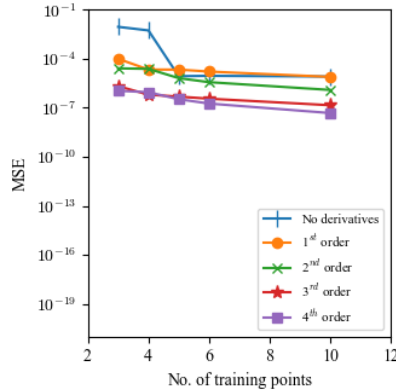


Figure 8: Results of numerical experiments for the predictive performance (measured by MSE) on 1D Griewank function for varying number of training points, order of derivatives, and increasing noise.

4. Dynamic Sparse GP Cholesky For Use In Digital Twin Application

In a DT system, surrogate models, like GPs, are employed to predict the target property of interest for the given state of the physical system. These surrogates are often trained with an initial set of data representing the physical system and then deployed in service as a DT. One of the critical aspects of DT is that the surrogate model should have the ability to be updated in a smooth fashion to accommodate any changes in the physical system. In other words, the surrogates need to be dynamically updated without the need for extensive re-training. Additionally, the state of the physical system can change significantly during the operation, and the trained surrogate may not perform well enough even with regular dynamic updates. Inevitably,

the surrogate model needs to be retrained. The DT system should have the ability to detect when to trigger re-training. Considering the above-discussed requirements for the surrogates in mind, we propose a dynamic sparse GP algorithm, which has the ability to be dynamically updated or retrained when new information is available.

4.1. Dynamic Derivative-Enhanced Sparse GP

Sparse GP offers a unique advantage for DT applications as it offers greater control over the location of entries in the matrix. This is where grouping by points offers a computational advantage because when new data with derivative information is available, new data can be included in the matrix by adding an additional column without re-computing the whole matrix. However, the major questions to be addressed are where the new column should be placed in the matrix and the sparsity of the new column. In order to address these questions, we rely on the idea of a dynamic supernode, which allows us only to re-evaluate the Cholesky factors of a small number of columns when building the matrix \mathbf{U} , thus eliminating the need for re-evaluating the complete sparse matrix.

Note that the dynamic supernode in this work should not be confused with the dynamic supernode concept used in matrix update and downdate [47]. In this section, we discuss two different approaches to generating and updating our dynamic supernodes. For the sake of simplicity, the approaches below are described for derivative-free measurements. However, they can be extended to include any arbitrary order of derivatives. We primarily describe the approaches below for noise-free measurements. However, they can be extended for noisy measurements by implementing the method mentioned in Section 3.3 to obtain the sparse Cholesky factor. Additionally, dynamic sparse GP is only applied for derivatives grouped by points, as it reduces the number of supernodes that need to be re-evaluated.

4.1.1. Supernode Update: Approach 1 (SU-Approach1)

Initially, we obtain \mathbf{P} from the MMD ordering mentioned in Sections 3.1 and 3.2. We generate supernodes, \mathcal{SN} , from the initial set of training points, \mathcal{D}_{train} . The obtained \mathbf{P} is split into two arrays, \mathbf{P}_{fix} and \mathbf{P}_{dyn} . The set \mathbf{P}_{dyn} is obtained by taking 20% of elements from the tail end of \mathbf{P} . In other words, the size, M , of \mathbf{P}_{dyn} is about 20% of N . After obtaining the fixed and dynamic sets, the supernodes with the parent set from \mathbf{P}_{fix} are considered fixed supernodes, \mathcal{SN}_{fix} . Alternatively, the supernodes with parents from \mathbf{P}_{dyn} are considered dynamic. In this approach, the parents and children of all the dynamic supernodes are merged together to form a single dynamic supernode, \mathcal{SN}_{dyn} . Doing so, we eliminate the need to pick the supernodes to which the new point will be added.

Once the fixed and dynamic sets are determined, the fixed supernodes \mathcal{SN}_{fix} would not be disturbed when an additional datapoint is available; only \mathcal{SN}_{dyn} is re-evaluated. Illustration of the above-discussed steps of creating fixed and dynamic dataset supernodes is shown in Fig. 9. On the left, we can see the $N \times N$ sparse matrix obtained using the initial ordering \mathbf{P} and \mathcal{SN} . The gray and orange columns in the middle figure show columns from the fixed and dynamic supernodes, respectively. Finally, all the supernodes with dynamic parents are merged to form \mathcal{SN}_{dyn} , shown as orange in the right-most figure in Fig. 9. If the derivative information is available as part of training dataset, \mathcal{D}_{train} , it can be added to \mathbf{P}_{fix} and \mathbf{P}_{dyn} , and subsequently \mathcal{SN}_{fix} and \mathcal{SN}_{dyn} are updated using the procedure described in Section 3.2.1.

When a new measurement is available, the generated dynamic supernode is updated in the following manner, and an illustration of the update process is shown in Fig. 10. When new data

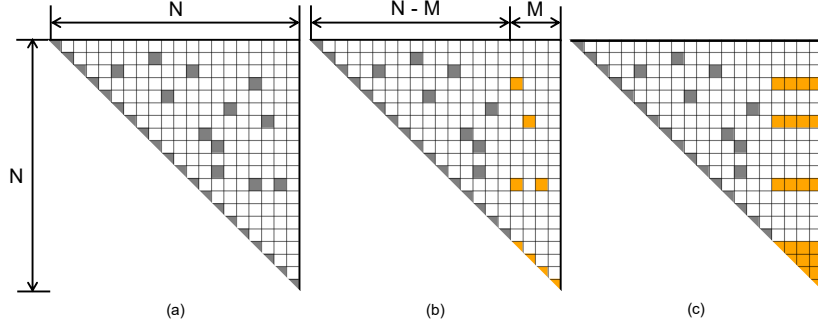


Figure 9: Illustration of generating fixed and dynamic supernodes to generate the sparse matrix. The diagonal entries shown in gray and orange correspond to parents of fixed supernodes, \mathcal{SN}_{fix} , and dynamic supernodes, \mathcal{SN}_{dyn} , respectively.

is available, it is added to the dynamic dataset \mathbf{P}_{dyn} , and the set is then reordered based on the MMD ordering scheme, and the parent of the \mathcal{SN}_{dyn} is updated based on the new order. The Cholesky factors of the updated \mathcal{SN}_{dyn} are evaluated and used to build the sparse matrix \mathbf{U} . This process is repeated until the model needs to be retrained. The criteria for re-training will be discussed in Section 4.2. When the model gets retrained, a new dynamic set, \mathbf{P}_{dyn} , is created as all the available data points, including the fixed set, are subjected to re-ordering.

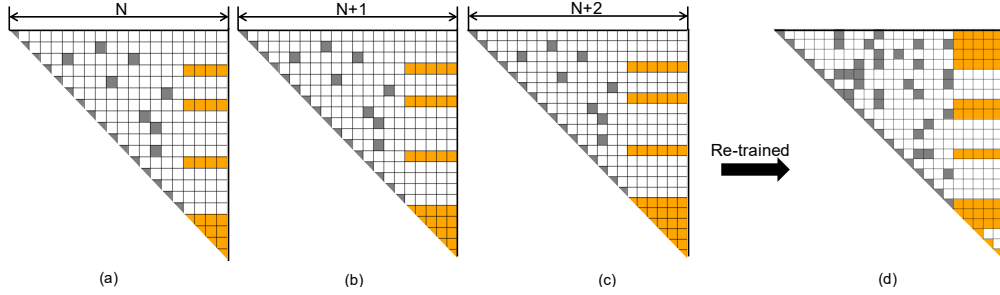


Figure 10: Illustration of the dynamic update of sparse GP using approach-1 with re-training. A $N \times N$ sparse matrix with fixed and dynamic set is shown in (a), new points are included in the dynamic supernodes (b and c), and finally, the model is re-trained, during which a new set of fixed and dynamic sets is created.

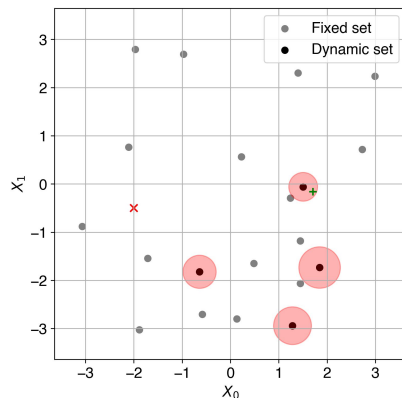


Figure 11: The figure shows the distribution of fixed (gray points) and dynamic set (black points) in a random dataset. When a new point (green point) falls within the radius ($\rho \cdot \mathbf{I}^{(i)}$), an existing supernode is updated. On the contrary, if any of the new point (red point) does not fall within the radius, a new supernode is created.

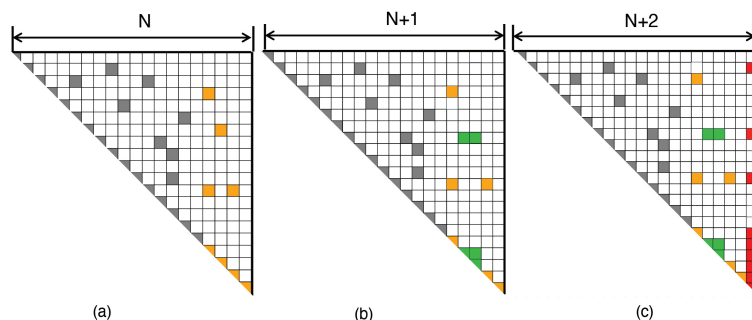


Figure 12: Illustration of the dynamic update of the GP using approach-2. On the left (a), the initial $N \times N$ sparse matrix with fixed and dynamic set highlighted in gray and orange, respectively. (b) Since the green point falls within the set radius, the dynamic supernode is updated, as shown in green. and (c) a supernode is created to accommodate a new point that does not fall within the set radius of any points in the dynamic set.

4.1.2. Supernode Update: Approach 2 (SU-Approach2)

In this approach, we obtain \mathbf{P}_{fix} and \mathbf{P}_{dyn} through $N - M$ and M split of \mathbf{P} , the same way as the previous approach. Once obtained, the supernodes with elements from \mathbf{P}_{fix} and \mathbf{P}_{dyn} are considered \mathcal{SN}_{fix} and \mathcal{SN}_{dyn} supernodes, respectively. Unlike the previous approach, the dynamic supernodes are not merged to form one dynamic supernode; in other words, multiple \mathcal{SN}_{dyn} can exist in this approach. Once the dynamic supernodes are formed during the initial training, the newly available points are added to one of the dynamic supernodes, or a new supernode is created, which is decided based on the geometric location of the new point with respect to the dynamic set. During the MMD ordering, each point has a length scale, $\mathbf{I}^{(i)}$, associated with it (Eq. 8), which is used along with ρ to form supernodes (see Section 3.1). When the newly available point falls within the radius ($\rho \cdot \mathbf{I}^{(i)}$) of any of the points in the dynamic set, then the new point is added to the \mathcal{SN}_{dyn} and the Cholesky factor is re-evaluated. If the new point does not fall within the radius of any of the points in the dynamic set, a new supernode is created. Figure 11

illustrates fixed points (gray), dynamic points (black), and a red circle shows the radius of the circle ($\rho \cdot \mathbf{I}^{(i)}$) for each of the dynamic points. If a newly available point (green) lies within the radius of any of the dynamic set, then the corresponding supernode is updated, and factors from other dynamic supernodes (and fixed) are reused. Visualization of this is provided in Fig. 12 (b). If the new point (red) does not fall within the specified radius of any of the points in the dynamic set, then a new supernode is created, red column in Fig. 12.

4.2. Fast Derivative-Enhanced Sparse GP Updating and Re-training

Algorithm 2 shows our fast derivative-enhanced sparse GP updating and re-training algorithm. The criteria for re-training are dependent on several factors. In this work, we choose three different criteria for re-training. First, when the state of the physical twin changes, the bounds of the input features may fall beyond the range of the previously trained DT surrogate. Therefore, it is crucial to identify the outliers when the new information is available from the physical twin. Secondly, when the sparse GP is dynamically updated, it may not result in a better surrogate than the deployed surrogate, i.e, the prediction error of the dynamically updated surrogate is more than the existing prediction error on a standard test dataset. In that case, the updated surrogate is not deployed, and the new data is stored and utilized while re-training. Additionally, a fixed budget can be set for the amount of unused new data, and re-training can be triggered when the budget is reached. Third, the deployed surrogate can be assumed to be diverging when the prediction error of the dynamically updated surrogate increases continuously for a set of newly added points. Therefore, a fixed number of continuous divergences is used to trigger re-training of the model. We now present the details of our algorithm.

Based on the initial set of data, \mathcal{D}_{train} , we train a surrogate model, $\mathcal{M}(\theta, \mathcal{D}_{train})$, which is a sparse GP with or without derivative information. The hyperparameters, θ , are optimized to reduce the prediction error, $L(\mathcal{M})$, on the test dataset, \mathcal{D}_{test} . The surrogate, \mathcal{M} , with optimized hyperparameters, is deployed as the digital twin. The prediction error from the deployed surrogate is set to L_{best} . The additional information obtained from the physical twin is added to a dynamic dataset, \mathcal{D}_{stream} , which will be used to dynamically update or re-train the deployed digital twin, \mathcal{M} . For every new point, \mathcal{D}_{new} , from the dynamic set, the algorithm checks whether one of the following criteria is met to trigger retraining. 1) Every x_{new} will be checked if it is an outlier compared to the existing dataset using the outlier detection algorithm (Algorithm B1 mentioned in Appendix B). The algorithm calculates the distance between the points used in the current state of GP using the k-nearest neighbor (k-NN) method. Based on the calculated distance, the threshold for outlier detection, τ , is determined using the hyperparameter, η_{out} . The new point, x_{new} , is classified as an outlier when the distance between the new point and existing points is greater than the calculated threshold, τ . 2) If the number of unused data points, η_{unused} , is greater than the preset budget, η_{budget} . 3) If the number of continuous divergence, η_{div} , is greater than the preset limit, η_{div_th} . If one of the above three criteria is met, then a complete re-training of the model is performed with a new X_{train} , which is a concatenation of the existing X_{train} and X_{stream} . Note that \mathcal{D}_{stream} is created for the sake of numerical experiments; in practice, whenever new data, \mathcal{D}_{new} , is available, it will be immediately used to update the model.

Algorithm 2 Fast sparse GP update with Outlier-based Retraining

```
1: Input: Initial training data  $\mathcal{D}_{train} = \{X_{train}, f_{train}\}$ , test data  $\mathcal{D}_{test}$ , stream of new data points  $X_{stream}$ 
2: Notation:  $\mathcal{M}(\theta, \mathcal{D})$  is a GP model with hyperparameters  $\theta$ .  $L(\mathcal{M})$  is the model's mean squared error on  $\mathcal{D}_{test}$ .

3: Initialize Model:
4:  $\mathcal{M} \leftarrow$  Train GP with  $\theta^*$  and  $\mathcal{D}_{train}$  ▷ Where  $\theta^*$  is optimized hyperparameters
5:  $L_{best} \leftarrow L(\mathcal{M})$ ,  $\eta_{div} = 0$ ,  $\eta_{unused} = 0$ 

6: for  $i = 1$  to  $|\mathcal{D}_{stream}|$  do
7:    $x_{new} \leftarrow X_{stream}[i]$ 
8:   if  $\text{IsOUTLIER}(x_{new}, X_{train})$  or  $\eta_{unused} > \eta_{budget}$  then or  $\eta_{div} > \eta_{div\_th}$ 
9:      $X_{add} \leftarrow \{x \in X_{stream}[1 \dots i] \mid x \notin X_{train}\}$ 
10:     $X_{train} \leftarrow X_{train} \cup X_{add}$  and update  $f_{train}$ 
11:     $\mathcal{M}(\theta_{new}^*) \leftarrow \text{FULLRETRAIN}(X_{train}, f_{train})$ 
12:    if  $L(\mathcal{M}(\theta_{new}^*)) < L_{best}$  then
13:       $\theta^* \leftarrow \theta_{new}^*$ 
14:    else
15:      repeat
16:         $\rho+ = 1$ 
17:         $\mathcal{M}(\theta_{new}^*) \leftarrow \text{FULLRETRAIN}(X_{train}, f_{train})$ 
18:         $\theta_{new}^* \leftarrow \arg \min_{\theta} L(\mathcal{M}(\theta, \mathcal{D}_{train}))$ 
19:        until  $L(\mathcal{M}(\theta_{new}^*)) < L_{best}$  or Sparsity of  $\mathcal{M}(\theta_{new}^*) >$  Lower-bound sparsity
20:      end if
21:       $\theta^* \leftarrow \theta_{new}^*$ 
22:       $\mathcal{M} \leftarrow$  Train GP with  $\theta^*$  and  $\mathcal{D}_{train}$ ,  $L_{best} \leftarrow L(\mathcal{M})$ 
23:    else
24:       $\mathcal{M}_{cand} \leftarrow \text{FASTUPDATE}(\mathcal{M}, x_{new})$  ▷ Dynamic update
25:       $L_{cand} \leftarrow L(\mathcal{M}_{cand})$ 
26:      if  $L_{cand} < L_{best}$  then
27:         $\mathcal{M} \leftarrow \mathcal{M}_{cand}$ ,  $L_{best} \leftarrow L_{cand}$  ▷ Accept the update
28:         $X_{train} \leftarrow X_{train} \cup \{x_{new}\}$  and update  $Y_{train}$ 
29:      else
30:         $\eta_{unused}+ = 1$  ▷ If update is not accepted,  $\mathcal{M}$  and  $L_{best}$  are unchanged.
31:        if  $L_{cand}/L_{best} > 1$  then ▷ Check for continuous divergence
32:           $\eta_{div}+ = 1$ 
33:        else
34:           $\eta_{div} = 0$ 
35:        end if
36:      end if
37:    end if
38: end for
```

During retraining, including additional datapoints does not guarantee an improved surrogate if the ρ is fixed. For a fixed ρ , the sparsity of the matrix increases as the number of training points

increases. If the prediction error from the retrained model is worse than the L_{best} , the ρ value is increased until the retrained model performs better or the sparsity of the retrained model does not fall below a set limit. If the re-training is not triggered, then the sparse GP is dynamically updated using one of the approaches discussed in Section 4.1.

4.3. Numerical Verification Experiments

The fast update algorithm with dynamic update using SU-Approach1 and SU-Approach2 is verified through numerical experiments (2D Griewank function) and is reported in Fig. 13. In the numerical experiments, we train the initial model using \mathcal{D}_{train} and it is updated using \mathcal{D}_{stream} through dynamic update or re-training based on criteria set in Algorithm 2. The size of \mathcal{D}_{train} and \mathcal{D}_{stream} are set at 25 and 10 points, respectively. Both datasets are randomly generated, and derivatives up to 4th-order are included in the dataset. The number of points added to the model from \mathcal{D}_{stream} is shown on the horizontal axis of Fig. 13, and the prediction error from the initially trained model is shown at 0 point. During training and update, the model is tested with the same dataset \mathcal{D}_{test} .

Figure 13 (a) shows the results of the experiment where the dynamic update is performed using SU-Approach1. Incorporating derivatives improved the prediction accuracy of the model. As additional data is included, the prediction error of the sparse GP is reduced noticeably, irrespective of the order of derivatives included in the training. Similar observations can be made from the results of SU-Approach2, as shown in Fig. 13 (b). Upon comparing the errors between the approaches, SU-Approach1 showed a lower prediction error than the error from SU-Approach2. This is due to the fact that for similar hyperparameters, SU-Approach1 exhibits lower sparsity than SU-Approach2, which is due to the formation of one big dynamic supernode by combining all smaller dynamic supernodes. Doing so increases the number of non-zero off-diagonal elements, leading to lower sparsity in SU-Approach1.

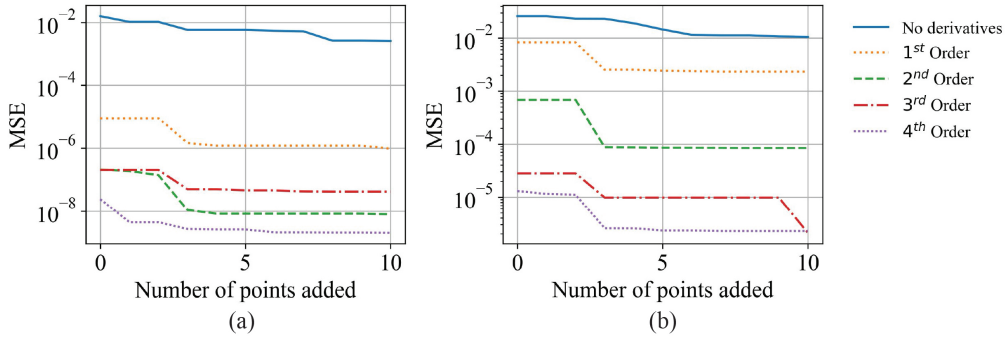


Figure 13: Results from dynamic update of the sparse GP using SU-Approach1 (a) and SU-Approach2 (b) tested on a 2D Griewank function with initial $\rho = 10$.

5. Application to Digital Twin

To demonstrate the derivative-enhanced sparse GP with the dynamic update algorithm, we apply it within a DT framework for predicting fatigue crack growth in an aircraft structure. In practice, fatigue cracks can evolve into arbitrary and complex shapes, but in many scenarios can be represented as a semi-elliptical surface crack in a thin plate, where cyclic load applied normal

to the crack faces drives fatigue crack growth. This geometry is illustrated in Fig.14, where a is the crack depth, c is the crack length on the surface of the plate, and t is the plate thickness. The rate at which the fatigue crack grows on the surface, $\frac{dc}{dN}$, is dependent on the material, applied cyclic loading, and state of the crack, defined by a and c . The objective is to update the DT model, using periodic inspections of the crack state. The updated DT model is then used to make improved, relative to the initial model, predictions about future crack states. Superscript PT and DT will be used to distinguish between measured values (from the physical twin) and sparse GP modeled values (from the digital twin), respectively.

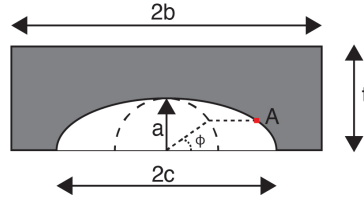


Figure 14: Semi-elliptical surface crack geometry in a finite plate, used to represent a thin aerospace component.

5.1. Digital Twin (DT) Workflow

The corresponding DT workflow in Figure 15 is a detailed version, specific to this application, of the general (abstract) workflow presented in Figure 1. Consistent with the general DT workflow, this application-specific workflow consists of three phases: initialization of the DT model (black outline), a dynamic update given PT observations (red outline), and a real-time tethering between the PT and DT for real-time prediction (green outline) of the crack state. We use subscript i for the dynamic update stage and j for the prognosis stage. In other words, i refers to inspection measurements of the crack state, which are used to update the DT model, while j refers to prediction of future crack states, given observations and model updates made at i .

The process involved in the initialization phase is marked in black dotted lines in Fig.15. This phase involves initial data acquisition and generating the initial DT , which is a sparse GP surrogate. The data used to train the DT model can come from lab-scale experiments (*e.g.*, nominal material information) or from similar PT s that are already in service (*e.g.*, other aircraft in a fleet). The obtained dataset, \mathcal{D} , includes the values of a , c , $\frac{dc}{dN}$, and derivatives of $\frac{dc}{dN}$ with respect to a and c up to an arbitrary order, d . Note that $\frac{dc}{dN}$ is referred to as f in Section 2.

\mathcal{D} is split into training data, \mathcal{D}_{train} , and testing data, \mathcal{D}_{test} , and used to develop the DT . For the model trained with d^{th} order, \mathcal{D}_{train} includes all the derivative terms up to order d , including mixed partial terms. See Section 5.3 for details on how these derivatives were obtained. The derivative values on the test dataset, \mathcal{D}_{test} , are not used, *i.e.*, the trained model is only tested for $\frac{dc}{dN}$ (*i.e.*, f). Upon completion of this initialization stage, it is important to note that the DT model is nominal in the sense that it does not yet capture any specific details of a particular PT . Once the initial DT model is trained and validated, it enters the service alongside the PT where it is updated to include PT -specific details.

During the inspection points, i , a measurement of the crack state (a and c) is obtained, which is used to update the DT model (either dynamically or via re-training) by correcting for any discrepancies between observed PT crack growth rate and corresponding DT model predictions. When the DT model is updated, a smooth transition is desired to avoid discontinuities, especially in cases where derivative information is included. It is important to note that in previous work in

this area, Bayesian updating methods were used, which was represented by a fixed model form and parameter re-calibration [11, 48]. In those prior cases, smoothness can be readily maintained. However, dynamically updating or retraining the model, as is done here, requires additional considerations for assessing smoothness. Immediately after initialization, differences between the PT observations and DT model predictions stem from the fact that the DT model is, at this point, a nominal model that is not yet specific to any particular PT. Initially, DT model updates would be adjusting for as-manufactured differences between a specific PT and the nominal case, for example. As flights (service) continue, updates would continue to update for additional PT-specific details that can include specific environments, material behavior, or changes in mechanisms. The newly available data from inspection, \mathcal{D}_{new} , is used to dynamically update or retrain the sparse GP surrogate using the algorithm 2, as outlined in red in Fig.15. Similar to \mathcal{D}_{train} , \mathcal{D}_{new} includes the values of a , c , $\frac{dc}{dN}$, and derivatives of $\frac{dc}{dN}$ with respect to a and c up to an arbitrary order, d , including all the mixed partial terms for $d > 1$.

During service, a real-time tethering between the PT and its DT is enabled for on-the-fly prognosis of the PT, outlined in green in Fig.15. The objective of this step is to predict crack growth in real-time, based on the most up-to-date DT model. At any real-time point, j , the DT model can be queried to predict the crack growth increment (Δc) based on the current state of the crack, expected load ($\Delta\sigma$), and expected number of cycles (ΔN). In this demonstration, the applied load is assumed to be the same throughout each PT flight. However, applied loading represents a DT model input variable that can be measured in real-time, j , (*i.e.*, unlike crack state, load measurement does not require an inspection at i) and, if measured, can be used to make improved DT predictions in this prognosis stage. Finally, to align this demonstration with practice, the DT prognosis steps (j) assume no new observations of crack state are made. Consequently, predictions are made using the updated DT model along with an assumption of self-similar crack growth: $\frac{a}{c}$ remains constant. This ratio, however, is updated at the inspection points, i , when the PT crack state is measured.

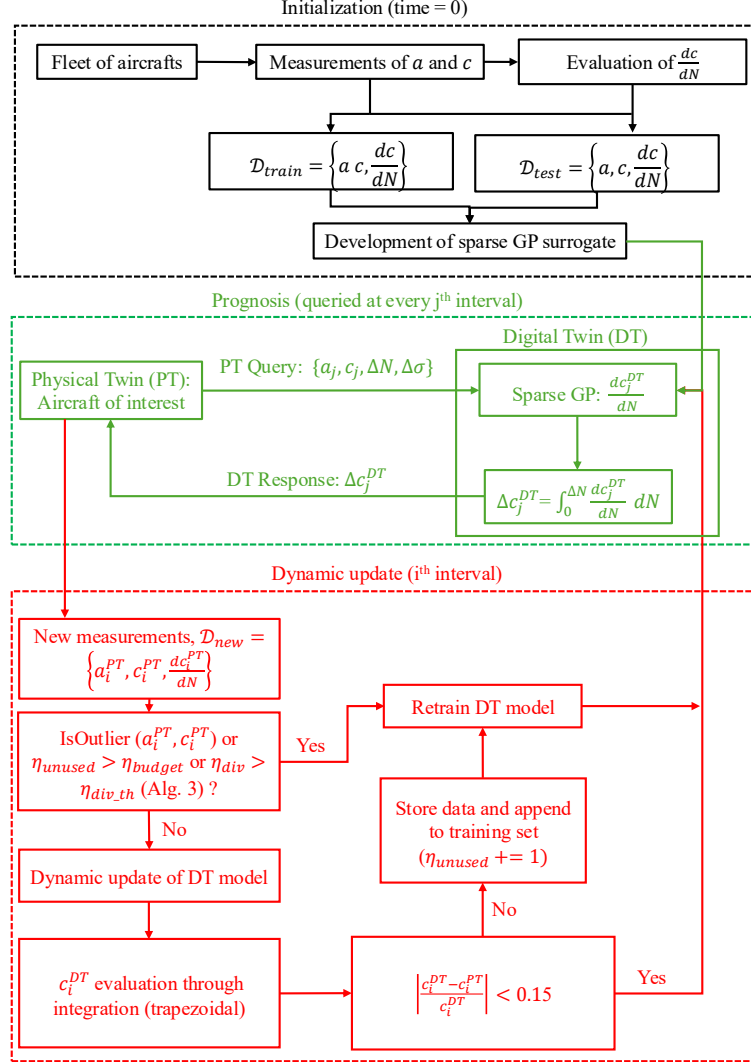


Figure 15: The employed DT workflow with initialization outlined in black, dynamic update of the DT model outlined in red, and real-time prognosis outlined in green. a^{PT} and c^{PT} correspond to the crack state obtained from the PT inspection. a^{DT} and c^{DT} represents the crack state prediction from the DT model.

5.2. Physical Twin (PT) Simulation

For this demonstration, no physical experiments were completed. Instead, a mechanics-based model was used to simulate PT flights. Assuming constant amplitude cyclic loading, the rate of fatigue crack growth is governed by the crack state (a and c), applied loading ($\Delta \sigma = \sigma_{max} - \sigma_{min}$), load ratio ($R = \frac{\sigma_{min}}{\sigma_{max}}$), and material properties (C and m). A load ratio, $R = 0$, was selected, implying $\sigma_{min} = 0$ and $\Delta \sigma = \sigma_{max}$, or simply σ (the subscript 'max' is dropped hereafter). Furthermore, the rate at which the fatigue crack grows (e.g., $\frac{dc}{dN}$ or $\frac{da}{dN}$) is assumed to follow the Paris law model:

$$\frac{dc}{dN} = C(\Delta K)^m, \quad (11)$$

where ΔK is cyclic stress intensity factor (described next), and for $R = 0$ the $\Delta K = K_{max}$. As with σ , we drop the subscript 'max' and refer to K hereafter. The stress intensity factor, K , defines the driving force for crack propagation and is a function of load and geometry variables (and not dependent on material properties). Stress intensity factors are generally obtained using high-fidelity computational fracture mechanics, see Ingraffea [49], or surrogate models that are most recently developed using a variety of machine learning approaches [50]. Here, we employ the stress intensity factor surrogate model, obtained via symbolic regression, by Merrell *et al.* [51]:

$$K = \sigma \cdot \sqrt{\frac{\pi l}{Q}} \cdot f_w\left(\frac{a}{t}, \frac{c}{b}\right) \cdot M\left(\frac{a}{t}, \frac{a}{c}\right) \cdot g\left(\frac{a}{t}, \frac{a}{c}, \phi\right), \quad (12)$$

where f_w is a finite width correction factor, M accounts for the aspect ratio (*i.e.*, $\frac{a}{c}$) of the crack, g accounts for the free surface effects, l is used to measure the perpendicular distance from the point of interest to the closest axis, and Q is the square of the complete elliptic integral of the second kind. Geometrical features a , c , t , b , and ϕ are defined in Figure 14. The equations defining f_w , M , and g can be found in Equations 18, 19, and 22 of Merrell *et al.* [51].

To simulate the PT crack evolution, an initial $a = 0.024$ and $c = 0.12$ ($\frac{a}{c} = 2$) was selected and inserted into a plate with dimensions $t = 0.1$ in and $b = 0.72$ in. Material properties $C = 5.25 \times 10^{-21}$ and $m = 3.97$ were sampled from a distribution mimicking aluminum 7075-T6 alloy. We refer to these C and m as C_2 and m_2 to distinguish from the nominal model obtained during the initialization step, which are referred to as C_1 and m_1 . Loading of $\sigma = 8500$ psi was then applied. With this PT, Equation 12 was used to simulate the stress intensity factor near the surface, $\phi = 5^\circ$, and depth, $\phi = 90^\circ$. With the K at each point (surface and depth), Equation 11 was then used to simulate the fatigue crack growth rate, which was integrated over N cycles to obtain the simulated PT crack state evolution, a^{PT} and c^{PT} . The crack state in the PT is inspected every $\Delta N = 5 \times 10^4$ cycles, which is then used to update the DT model. The simulated service life of the PT is defined as $N = 7.5 \times 10^5$ cycles.

5.3. Digital Twin (DT) Model Setup

The initial DT model was trained within $0.001 \leq a \leq 0.08$ inch and $0.2 \leq \frac{a}{c} \leq 2$ inch. This domain is defined such that the ranges of $\frac{a}{c}$, $\frac{a}{t}$, and $\frac{c}{b}$ remain within the valid bounds for K , per Eq.12. Pairwise a and c data were then obtained by sampling 10 and 500 points, respectively, from uniform distributions. The nominal DT model was then defined to consist of a nominal Aluminum alloy with $C_1 = 5.52 \times 10^{-21}$ and $m_1 = 4$ [52]. Equation 11 was then used to compute the corresponding $\frac{dc}{dN}$ for each (a, c) pair to form \mathcal{D}_{train} (size 10) and \mathcal{D}_{test} (size 500), *i.e.*, representing a nominal prior dataset. Effectively, the differences of $C_2 < C_1$ and $m_2 < m_1$ result in PT crack growth that is significantly slower than the nominal case.

Obtaining the derivatives of $\frac{dc}{dN}$ with respect to a and c , to be included in \mathcal{D}_{train} (see Section 5.1), could be obtained using numerical differentiation of the existing $(a, c, \frac{dc}{dN})$ data. This approach would likely be most representative of in-practice cases, wherein the $(a, c, \frac{dc}{dN})$ data would be measured, while higher-order derivatives would likely not be possible to measure directly. However, numerical differentiation would introduce (well understood) error into this process and

potentially obfuscate the desired assessment of the algorithm efficacy. Consequently, the derivatives in this application problem are obtained analytically by taking all derivatives of Equation 11 with respect to a and c . It is left to future work to demonstrate the effect of noisy derivative data, using algorithms presented in Section 3.5.

Using the nominal datasets, an exact GP (no sparsification, see Section 2), and a dynamic sparse GP model with an initial ρ of 20 are trained, providing a baseline comparison for the sparse GP performance. For all DT models (whether exact or sparse), a small value of jitter is added along the diagonal to improve the conditioning of the matrix. Once trained, we use the DT model to predict the crack growth rate of the crack in the PT. At every i (inspection) step, the sparse GP model is either re-trained or dynamically updated depending on the re-training criteria in Algorithm 2, and the SU-Approach1 for the dynamic update is used due to the improved prediction accuracy observed in the numerical experiments.

Finally, the objective of DT model updating is to provide individualized predictions of the PT, which implies evolving away from the initial nominal DT model, as necessary. To quantitatively assess this objective, we complete a parallel study in which the initialized exact GP model (DT) remains fixed during the simulated service life (*i.e.*, the initial DT model is not updated with new crack state observations). Then, during prognosis steps, DT model predictions (prognosis) are made from the observed crack state (a^{PT} and c^{PT}). In doing so, we are able to report on efficacy of the DT model updating and retraining, specifically, while keeping all other variables fixed.

5.4. Results

Figure 16 shows the predicted crack growth rate (left column) and the relative percent difference, η , between the predicted (DT) and actual (PT) crack growth rates (right column) is given by:

$$\eta = \left| \frac{\frac{dc^{PT}}{dN} - \frac{dc^{DT}}{dN}}{\frac{dc^{PT}}{dN}} \right| \cdot 100. \quad (13)$$

Each plot in Figure 16 illustrates results for the DT model trained with orders of derivatives ranging from zeroth to third. The orange and blue points indicate the crack growth rate observed in the initial nominal DT model and in the PT inspections, respectively. The vertical dotted lines represent the inspection steps, i , where the first vertical dashed line corresponds to the first inspection and update point, $i = 1$. At $i = 0$ (at $N = 0$, not shown), quality control inspection data could be acquired after manufacturing but before the PT service life. At this point, crack state data (a^{PT} and c^{PT}) could be obtained but PT-specific crack growth rate data would not be available until $i = 1$. Consequently, since the DT model has not yet been updated to account for the any specific PT at $i = 0$, the corresponding initial DT model prediction will be that of a nominal crack growth rate (orange dots) until the first update, $i = 1$. Training of the initial nominal model benefited significantly from including derivatives in the training data, which reduced the DT model error from 40% to less than 10% for the DT model with 1st-order derivatives and higher.

Figure 16(a) plots the results of the baseline case in which the initial exact GP model was not updated throughout the DT model service life. In this baseline case, it was expected that the DT model will track the nominal (initial) data and not evolve toward the PT data. As expected, plotting η over the service life illustrates no improvement and eventual divergence in accuracy with respect to the PT. In other words, the initial nominal model becomes increasingly inaccurate

as the PT service life progresses. Additionally, the DT models that were trained using at least first-order derivatives more accurately model the nominal crack growth rate.

The left column of Figure 16(a) illustrates a significant underprediction of the fatigue crack growth rate for the DT model trained without derivative data. It may be misleading that this ‘No derivatives’ data aligns with the PT data; however, in this baseline case, the DT model was not updated based on PT data. Instead, any apparent alignment of these data is purely coincidental, in that the ‘No derivatives’ case underpredicts (erroneously) the nominal data and the specific PT used in this study happens to have a reduced crack growth rate with respect to the nominal case (recall Section 5.3). This same observations is made in Figures 16(b) and 16(c). Because of this potentially misleading result, plots of η in the right column of Figure 16 do not include the “No derivatives” datasets.

When the DT model is updated at each i step, as in Figures 16(b) and 16(c), the prediction from the DT model becomes increasingly accurate as the PT service life progresses. This is evident by comparing η values along the right side of Figure 17. For the exact GP baseline case, Figure 16(b), η at $i = 1$ was 40%, 29%, and 26% when trained with 1st-order, 2nd-order, and 3rd-order, respectively. Dynamic updating of the exact GP model at subsequent i steps reduced this discrepancy to 21%, 21%, and 18% for 1st-order, 2nd-order, and 3rd-order, respectively. Similarly, for the dynamic sparse GP model, the updates helped improve the accuracy with η reducing from 40%, 30%, and 26% to 26%, 24%, and 21% for 1st-order, 2nd-order, and 3rd-order, respectively.

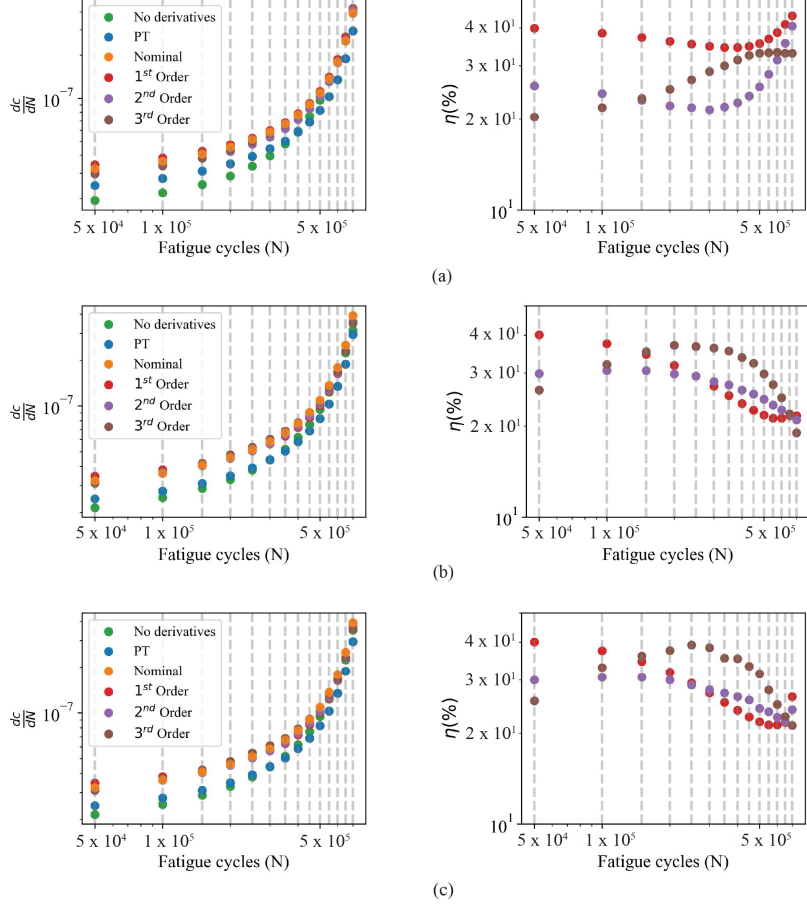


Figure 16: Crack growth rate predicted using (a) nominal DT, (b) exact GP with re-training at every i step, and (c) dynamic sparse GP with initial ρ of 20.

For both exact GP with re-training, Figure 16(b), and dynamic sparse GP, Figure 16(c), the prediction from the DT model that was trained with 1st-order derivatives shows an increasing η at $N \gtrsim 5 \times 10^5$ cycles. It is at this cycle count that the $\frac{a}{c}$ ratio of the crack state changes from greater than 1 to less than 1. Due to the uniform random sampling employed at initialization, the initial training dataset (\mathcal{D}_{train}) comprised of less than 20% where $\frac{a}{c} < 1$ with a mean $\frac{a}{c}$ of 1.45. Interestingly, however, for cases in which the DT model was trained with same initial points and with up to 3rd-order derivatives, η decreased with increasing N , and the switch from $\frac{a}{c}$ did not have such an effect on the model. This suggests that even with a limited dataset, including higher-order derivatives can help improve accuracy in cases of low training data (in this case, just 10 data points). An additional experiment with 20 initial training points and a wider range of $\frac{a}{c}$ demonstrated a decreasing η with increasing N with 1st-order derivatives, which is not shown here for the sake of brevity.

Additional experiments were performed to quantify the effect of dynamic model update frequency. Figure 17 shows η for the sparse GP using update intervals of 1×10^4 cycles and 5×10^4

cycles. Clearly, and as expected, the DT model accuracy improves with increased model update frequency.

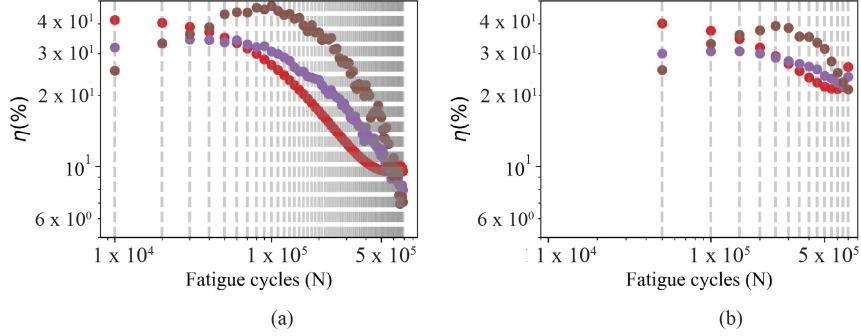


Figure 17: Percent difference in predicted crack growth rate compared to the PT with update intervals of (ΔN) (a) 1×10^4 cycles and (b) 5×10^4 cycles.

6. Summary and Conclusions

In this work, the accuracy of DT models is improved by incorporating higher-order derivatives into the training data. As it is well-suited for DT applications, we present a dynamic updating algorithm for sparse GP models to utilize the updated data and corresponding derivatives from a PT. The results of this study are summarized below:

- Numerical experiments on GPs with derivatives showed a significant reduction in prediction error. For a 3D Griewank function with 27 points, GP models trained with 4th-order derivatives showed prediction error on the order of 10^{-13} when compared to 10^{-3} for GP models trained without derivatives. Although incorporating derivatives improves the model prediction accuracy, it comes at the expense of an increased size of the covariance matrix.
- We leverage a sparse GP algorithm that utilizes maximum-minimum ordering and an aggregated sparsity set. This algorithm is then generalized to incorporate derivatives in the training data. The numerical experiments showed that increasing the order of derivative improved the prediction accuracy of the sparse GP model. For sufficiently larger ρ , the sparse GP has shown similar prediction accuracy to that of the exact GP.
- We develop and present two different dynamic update algorithms, which enable new data to be added to the sparse GP without requiring complete retraining. Whenever new information is available, it is added to the dynamic supernode and only the Cholesky factors of the dynamic supernode are reevaluated; remaining factors are reused. Such a dynamic update offers a significant computational advantage as it eliminates the need for full matrix evaluation. Numerical experiments showed that the prediction accuracy of the sparse GP model improved when new data were added to the sparse matrix.
- Finally, we apply the developed derivative-informed dynamic sparse GP algorithm to a fatigue crack growth DT problem. Similar to the simpler numerical experiments, incorporating derivatives was observed to significantly increase the DT model prediction accuracy.

Additionally, the dynamic update of the sparse GP (DT model) throughout the simulated service life demonstrated the ability to individualize initially nominal predictions to that of a specific PT. Without such updates, predictions from the initial nominal DT model diverged from the PT behavior, while increasing the DT model update frequency continually improved the DT model accuracy.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- [1] Eric J Tuegel, Anthony R Ingraffea, Thomas G Eason, and S Michael Spottswood. Reengineering aircraft structural life prediction using a digital twin. *International Journal of Aerospace Engineering*, 2011(1):154798, 2011.
- [2] Edward Glaessgen and David Stargel. The digital twin paradigm for future nasa and us air force vehicles. In *53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA*, page 1818, 2012.
- [3] Michael W. Grieves. *Digital Twins: Past, Present, and Future*, pages 97–121. Springer International Publishing, Cham, 2023. ISBN 978-3-031-21343-4.
- [4] Michael G Kapteyn, David J Knezevic, DBP Huynh, Minh Tran, and Karen E Willcox. Data-driven physics-based digital twins via a library of component-based reduced-order models. *International Journal for Numerical Methods in Engineering*, 123(13):2986–3003, 2022.
- [5] Albert Cerrone, Jacob Hochhalter, Gerd Heber, and Anthony Ingraffea. On the effects of modeling as-manufactured geometry: Toward digital twin. *International Journal of Aerospace Engineering*, 2014(1):439278, 2014.
- [6] Federal Aviation Administration. Advisory Circular 23-13A: Fatigue, Fail-Safe, and Damage Tolerance Evaluation of Metallic Structure for Normal, Utility, Acrobatic, and Commuter Category Airplanes. https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_23-13A.pdf, 2005. Accessed: 2025-10-21.
- [7] Federal Aviation Administration. Advisory Circular 25.571-1D: Damage Tolerance and Fatigue Evaluation of Structure. https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_25_571-1D_.pdf, 2011. Accessed: 2025-10-21.
- [8] Min Liao, Guillaume Renaud, and Yan Bombardier. Airframe digital twin technology adaptability assessment and technology demonstration. *Engineering Fracture Mechanics*, 225:106793, 2020. ISSN 0013-7944. doi: <https://doi.org/10.1016/j.engfracmech.2019.106793>. URL <https://www.sciencedirect.com/science/article/pii/S0013794419303182>.

- [9] Harry Millwater, Juan Ocampo, and Nathan Crosby. Probabilistic methods for risk assessment of airframe digital twin structures. *Engineering Fracture Mechanics*, 221:106674, 2019. ISSN 0013-7944. doi: <https://doi.org/10.1016/j.engfracmech.2019.106674>. URL <https://www.sciencedirect.com/science/article/pii/S0013794419302966>.
- [10] Gary Whelan and David L. McDowell. Uncertainty quantification in icme workflows for fatigue critical computational modeling. *Engineering Fracture Mechanics*, 220:106673, 2019. ISSN 0013-7944. doi: <https://doi.org/10.1016/j.engfracmech.2019.106673>. URL <https://www.sciencedirect.com/science/article/pii/S0013794419304497>.
- [11] Saikumar R. Yeratapally, Patrick E. Leser, Jacob D. Hochhalter, William P. Leser, and Timothy J. Ruggles. A digital twin feasibility study (part i): Non-deterministic predictions of fatigue life in aluminum alloy 7075-t651 using a microstructure-based multi-scale model. *Engineering Fracture Mechanics*, 228:106888, 2020. ISSN 0013-7944. doi: <https://doi.org/10.1016/j.engfracmech.2020.106888>. URL <https://www.sciencedirect.com/science/article/pii/S0013794419307982>.
- [12] Patrick E. Leser, James E. Warner, William P. Leser, Geoffrey F. Bomarito, John A. Newman, and Jacob D. Hochhalter. A digital twin feasibility study (part ii): Non-deterministic predictions of fatigue life using in-situ diagnostics and prognostics. *Engineering Fracture Mechanics*, 229:106903, 2020. ISSN 0013-7944. doi: <https://doi.org/10.1016/j.engfracmech.2020.106903>. URL <https://www.sciencedirect.com/science/article/pii/S0013794419307441>.
- [13] Yifan Chen, Houman Owhadi, and Florian Schäfer. Sparse Cholesky Factorization for Solving Nonlinear PDEs via Gaussian Processes, March 2024. URL <http://arxiv.org/abs/2304.01294>. arXiv:2304.01294 [math].
- [14] Shikai Fang, Madison Cooley, Da Long, Shibo Li, Robert Kirby, and Shandian Zhe. Solving High Frequency and Multi-Scale PDEs with Gaussian Processes. *The Twelfth International Conference on Learning Representations (ICLR 2024)*, March 2024. doi: 10.48550/arXiv.2311.04465. URL <https://arxiv.org/abs/2311.04465>. arXiv:2311.04465 [cs.LG].
- [15] E Solak, R Murray-smith, W E Leithead, D J Leith, and Carl E Rasmussen. Derivative Observations in Gaussian Process Models of Dynamic Systems.
- [16] Anqi Wu, Mikio C. Aoi, and Jonathan W. Pillow. Exploiting gradients and Hessians in Bayesian optimization and Bayesian quadrature, March 2018. URL <http://arxiv.org/abs/1704.00060>. arXiv:1704.00060 [stat].
- [17] Hongqiao Wang and Xiang Zhou. Explicit Estimation of Derivatives from Data and Differential Equations by Gaussian Process Regression, October 2020. URL <http://arxiv.org/abs/2004.05796>. arXiv:2004.05796 [stat].
- [18] David Eriksson, Kun Dong, Eric Hans Lee, David Bindel, and Andrew Gordon Wilson. Scaling Gaussian Process Regression with Derivatives, October 2018. URL <http://arxiv.org/abs/1810.12283>. arXiv:1810.12283 [cs].
- [19] Misha Padidar, Xinran Zhu, and Leo Huang. Scaling Gaussian Processes with Derivative Information Using Variational Inference.

- [20] Soham Mukherjee, Manfred Claassen, and Paul-Christian Bürkner. DGP-LVM: Derivative Gaussian process latent variable models. *Statistics and Computing*, 35(5):120, October 2025. ISSN 0960-3174, 1573-1375. doi: 10.1007/s11222-025-10644-4. URL <http://arxiv.org/abs/2404.04074>. arXiv:2404.04074 [stat].
- [21] Matthew Dowling. Hida-matérn kernel hida-matérn kernel. 2021. URL <https://api.semanticscholar.org/CorpusID:245539494>.
- [22] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634. URL <https://www.tandfonline.com/doi/abs/10.1080/00401706.1970.10488634>.
- [23] Lisa Schönenberger and Hans-Georg Beyer. Success rate of evolution strategies on the multimodal griewank function. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2024. doi: 10.1109/CEC60901.2024.10612209.
- [24] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian Process Meets Big Data: A Review of Scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4405–4423, November 2020. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2019.2957109. URL <https://ieeexplore.ieee.org/document/8951257/>.
- [25] Matthew J. Heaton, Abhirup Datta, Andrew O. Finley, Reinhard Furrer, Joseph Guinness, Rajarshi Guhaniyogi, Florian Gerber, Robert B. Gramacy, Dorit Hammerling, Matthias Katzfuss, Finn Lindgren, Douglas W. Nychka, Furong Sun, and Andrew Zammit-Mangion. A Case Study Competition Among Methods for Analyzing Large Spatial Data. *Journal of Agricultural, Biological and Environmental Statistics*, 24(3):398–425, September 2019. ISSN 1085-7117, 1537-2693. doi: 10.1007/s13253-018-00348-w. URL <http://link.springer.com/10.1007/s13253-018-00348-w>.
- [26] Cameron Musco and Christopher Musco. Recursive Sampling for the Nyström Method, November 2017. URL <http://arxiv.org/abs/1605.07583>. arXiv:1605.07583 [cs].
- [27] Yifan Chen, Ethan N. Epperly, Joel A. Tropp, and Robert J. Webber. Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations. *Communications on Pure and Applied Mathematics*, 78(5):995–1041, May 2025. ISSN 0010-3640, 1097-0312. doi: 10.1002/cpa.22234. URL <https://onlinelibrary.wiley.com/doi/10.1002/cpa.22234>.
- [28] A. V. Vecchia. Estimation and Model Identification for Continuous Spatial Processes. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 50(2):297–312, January 1988. ISSN 1369-7412, 1467-9868. doi: 10.1111/j.2517-6161.1988.tb01729.x. URL <https://academic.oup.com/jrsssb/article/50/2/297/7027819>.
- [29] Reinhard Furrer, Marc G Genton, and Douglas Nychka. Covariance Tapering for Interpolation of Large Spatial Datasets. *Journal of Computational and Graphical Statistics*, 15(3):502–523, September 2006. ISSN 1061-8600, 1537-2715. doi: 10.1198/106186006X132178. URL <http://www.tandfonline.com/doi/abs/10.1198/106186006X132178>.

- [30] Abhirup Datta, Sudipto Banerjee, Andrew O. Finley, and Alan E. Gelfand. Hierarchical Nearest-Neighbor Gaussian Process Models for Large Geostatistical Datasets. *Journal of the American Statistical Association*, 111(514):800–812, April 2016. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.2015.1044091. URL <https://www.tandfonline.com/doi/full/10.1080/01621459.2015.1044091>.
- [31] Abhirup Datta, Sudipto Banerjee, Andrew O. Finley, Nicholas A. S. Hamm, and Martijn Schaap. Nonseparable dynamic nearest neighbor Gaussian process models for large spatio-temporal data with an application to particulate matter analysis. *The Annals of Applied Statistics*, 10(3), September 2016. ISSN 1932-6157. doi: 10.1214/16-AOAS931. URL <https://projecteuclid.org/journals/annals-of-applied-statistics/volume-10/issue-3/Nonseparable-dynamic-nearest-neighbor-Gaussian-process-models-for-large-spatio-temporal-data-with-an-application-to-particulate-matter-analysis/10.1214/16-AOAS931.full>.
- [32] Alex Smola and Peter Bartlett. Sparse greedy gaussian process regression. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.
- [33] Jian Cao, Myeongjong Kang, Felix Jimenez, Huiyan Sang, Florian Schafer, and Matthias Katzfuss. Variational sparse inverse Cholesky approximation for latent Gaussian processes via double Kullback-Leibler minimization, May 2023. URL <http://arxiv.org/abs/2301.13303>. arXiv:2301.13303 [stat].
- [34] Ali Rahimi and Ben Recht. Random Features for Large-Scale Kernel Machines.
- [35] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005. URL <http://jmlr.org/papers/v6/quinonero-candela05a.html>.
- [36] Michael L. Stein, Zhiyi Chi, and Leah J. Welty. Approximating Likelihoods for Large Spatial Data Sets. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 66(2):275–296, May 2004. ISSN 1369-7412, 1467-9868. doi: 10.1046/j.1369-7412.2003.05512.x. URL <https://academic.oup.com/jrsssb/article/66/2/275/7098431>.
- [37] Ying Sun and Michael L. Stein. Statistically and Computationally Efficient Estimating Equations for Large Spatial Datasets. *Journal of Computational and Graphical Statistics*, 25(1):187–208, January 2016. ISSN 1061-8600, 1537-2715. doi: 10.1080/10618600.2014.975230. URL <https://www.tandfonline.com/doi/full/10.1080/10618600.2014.975230>.
- [38] Matthias Katzfuss, Joseph Guinness, and Earl Lawrence. Scaled Vecchia approximation for fast computer-model emulation, July 2021. URL <http://arxiv.org/abs/2005.00386>. arXiv:2005.00386 [stat].
- [39] Stephen Huan, Joseph Guinness, Matthias Katzfuss, Houman Owhadi, and Florian Schäfer. Sparse inverse Cholesky factorization of dense kernel matrices by greedy conditional selection, May 2025. URL <http://arxiv.org/abs/2307.11648>. arXiv:2307.11648 [stat].

- [40] Joseph Guinness. Permutation and Grouping Methods for Sharpening Gaussian Process Approximations. *Technometrics*, 60(4):415–429, October 2018. ISSN 0040-1706, 1537-2723. doi: 10.1080/00401706.2018.1437476. URL <http://arxiv.org/abs/1609.05372>. arXiv:1609.05372 [stat].
- [41] Myeongjong Kang and Matthias Katzfuss. Correlation-based sparse inverse Cholesky factorization for fast Gaussian-process inference. *Statistics and Computing*, 33(3):56, June 2023. ISSN 0960-3174, 1573-1375. doi: 10.1007/s11222-023-10231-5. URL <http://arxiv.org/abs/2112.14591>. arXiv:2112.14591 [stat].
- [42] Matthias Katzfuss and Joseph Guinness. A General Framework for Vecchia Approximations of Gaussian Processes. *Statistical Science*, 36(1), February 2021. ISSN 0883-4237. doi: 10.1214/19-STS755. URL <https://projecteuclid.org/journals/statistical-science/volume-36/issue-1/A-General-Framework-for-Vecchia-Approximations-of-Gaussian-Processes/10.1214/19-STS755.full>.
- [43] Florian Schäfer and Houman Owhadi. Sparse Recovery of Elliptic Solvers from Matrix-Vector Products. *SIAM Journal on Scientific Computing*, 46(2):A998–A1025, April 2024. ISSN 1064-8275, 1095-7197. doi: 10.1137/22M154226X. URL <https://epubs.siam.org/doi/10.1137/22M154226X>.
- [44] Florian Schäfer, Matthias Katzfuss, and Houman Owhadi. Sparse Cholesky Factorization by Kullback–Leibler Minimization. *SIAM Journal on Scientific Computing*, 43(3):A2019–A2046, January 2021. ISSN 1064-8275, 1095-7197. doi: 10.1137/20M1336254. URL <https://epubs.siam.org/doi/10.1137/20M1336254>.
- [45] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M. Stuart. Solving and Learning Nonlinear PDEs with Gaussian Processes, August 2021. URL <http://arxiv.org/abs/2103.12959>. arXiv:2103.12959 [math].
- [46] Florian Schäfer, T. J. Sullivan, and Houman Owhadi. Compression, Inversion, and Approximate PCA of Dense Kernel Matrices at Near-Linear Computational Complexity. *Multiscale Modeling & Simulation*, 19(2):688–730, 2021. doi: 10.1137/19M129526X. URL <https://doi.org/10.1137/19M129526X>.
- [47] Timothy A. Davis and William W. Hager. Dynamic Supernodes in Sparse Cholesky Update/Downdate and Triangular Solves. *ACM Transactions on Mathematical Software*, 35(4):1–23, February 2009. ISSN 0098-3500, 1557-7295. doi: 10.1145/1462173.1462176. URL <https://dl.acm.org/doi/10.1145/1462173.1462176>.
- [48] Patrick E. Leser, James E. Warner, William P. Leser, Geoffrey F. Bomarito, John A. Newman, and Jacob D. Hochhalter. A digital twin feasibility study (Part II): Non-deterministic predictions of fatigue life using in-situ diagnostics and prognostics. *Engineering Fracture Mechanics*, 229:106903, April 2020. ISSN 00137944. doi: 10.1016/j.engfracmech.2020.106903. URL <https://linkinghub.elsevier.com/retrieve/pii/S0013794419307441>.
- [49] Anthony R Ingraffea. Computational fracture mechanics. *Encyclopedia of computational mechanics*, 2004.

- [50] Tushar Gautam, Jacob Hochhalter, Shandian Zhe, Eric Lindgren, and Robert M. Kirby. Developing robust stress intensity factor models using fourier-based data analysis to guide machine learning method selection and training. *Engineering Fracture Mechanics*, 326:111387, 2025. ISSN 0013-7944. doi: <https://doi.org/10.1016/j.engfracmech.2025.111387>. URL <https://www.sciencedirect.com/science/article/pii/S0013794425005880>.
- [51] Jonas Merrell, John Emery, Robert M. Kirby, and Jacob Hochhalter. Stress intensity factor models using mechanics-guided decomposition and symbolic regression. *Engineering Fracture Mechanics*, 310:110432, 2024. ISSN 0013-7944. doi: <https://doi.org/10.1016/j.engfracmech.2024.110432>. URL <https://www.sciencedirect.com/science/article/pii/S0013794424005952>.
- [52] C.Michael Hudson and Joseph T. Scardina. Effect of stress ratio on fatigue-crack growth in 7075-t6 aluminum-alloy sheet. *Engineering Fracture Mechanics*, 1(3):429–446, 1969. ISSN 0013-7944. doi: [https://doi.org/10.1016/0013-7944\(69\)90003-4](https://doi.org/10.1016/0013-7944(69)90003-4). URL <https://www.sciencedirect.com/science/article/pii/0013794469900034>.

APPENDIX A. ORDERING ALGORITHMS

A.1. Additional Details Regarding the Ordering Algorithms Discussed in Section 3

This section provides details on the “point-wise ordering algorithm 2”, the “measurement-wise ordering algorithm 1”, and the “measurement-wise ordering algorithm 2”. Additionally, a detailed comparison between the predictive performance of these three methods is provided in contrast to the predictive performance of “point-wise ordering algorithm 1” presented in Section 3 of this work.

In measurement-wise ordering algorithm 1, the derivative measurements are grouped separately. For better understanding, we illustrate the structure of the covariance matrix that has function value $f(x)$ and its corresponding first-order derivative measurement $\nabla f(x)$ in Fig.A.1.1. The plot shows the structure of the covariance matrix when all the derivative-free measurements, f , are ordered first, followed by ∇f , i.e., $\mathbf{F} = [f^{\mathbf{P}(1):\mathbf{P}(N)}, \nabla f^{\mathbf{P}(1):\mathbf{P}(N)}]$. For exact GP, the ordering should not have any effect on the distribution, as this method simply permutes the same elements within the matrix.

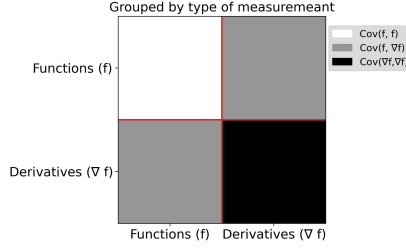


Figure A.1.1: The figure illustrates the measurement-wise ordering algorithm 1 of incorporating derivative measurement in the formation of the \mathbf{K}_{der} matrix. The derivative measurements are placed after the functional observation, following the same ordering.

For this method, we obtain the initial ordering \mathbf{P} using Eq.7 without any derivative measurements, and then we extend \mathbf{P} to incorporate derivative measurements to obtain \mathbf{P}^d . A subscript $me-1$ is added to \mathbf{P}^d to refer to the ordering grouped by the measurement-wise ordering algorithm 1. The algorithm to obtain \mathbf{P}_{me-1}^d is shown in Algorithm A1.

Algorithm A1 Constructing the \mathbf{P}_{me-1}^d array

- 1: **Input:** \mathbf{P} from MMD ordering
 - 2: **Output:** \mathbf{P}_{me-1}^d
 - 3: $td \leftarrow \lfloor N_d/N \rfloor$
 - 4: **for** $b \leftarrow 1$ to td **do**
 - 5: $offset \leftarrow b \cdot N$
 - 6: **for** $k \leftarrow 1$ to N **do**
 - 7: $\mathbf{P}_{me-1}^d[offset + k] \leftarrow \mathbf{P}[k] + offset$
 - 8: **end for**
 - 9: **end for**
-

To obtain the \mathbf{P}_{me-1}^d , the functional observations are first placed as per the MMD ordering, followed by all the derivative observations, following the same ordering. In other words, each derivative measurement is ordered the same as \mathbf{P} , and is stacked to \mathbf{P}_{me-1}^d .

\mathcal{SN} are originally obtained without any derivative measurement, through the procedure described in Section 3.1, and then they are generated to include derivative measurements in them. This is done by generating a new set of parents and children that corresponds to the indices of the derivative measurements, and they are added to the existing supernode, \mathcal{SN} , to obtain \mathcal{SN}_{me-1}^d . \mathcal{SN}_{me-1}^d is a list of multiple supernodes that are used to build the sparse matrix.

We perform experiments by varying ρ values to compare the prediction errors between the measurement-wise ordering algorithm 1 and the point-wise ordering algorithm 1. The results are reported in Fig.A.1.2. For both the groupings, increasing the order of derivatives reduces the prediction error when the ρ is sufficiently large, let's call it the saturation point ρ_s , which is dependent on the number of training points. At ρ_s , the sparsity of the matrix \mathbf{U} reaches the lower bound, and any further increase in ρ is not expected to have a significant effect on prediction. When the number of training points is 16, the prediction error plateaus after $\rho = 4$, and any further increase in ρ does not result in improved prediction accuracy. When N is 36 and 64, the value of ρ_s is 5 and 8, respectively. When the $\rho < \rho_s$, the prediction error increases almost linearly in log scale with a decrease in ρ . This is because when the matrix becomes sparse, the information of specific point measurements is lost; thus, the model is expected to have a higher prediction error. Interestingly, the increase in error is much more significant when the derivatives are included in training, as noticed by differences in slope for different orders of derivatives. When the matrix becomes increasingly sparse, we lose the information of points along with their derivatives. We know that the model prediction error is reduced significantly when derivatives are included. On the contrary, we are expected to lose accuracy significantly when some derivative information is lost in the sparse. Similar observations can be made for the results shown in Fig.A.1.3 of the 1D Griewank function as well.

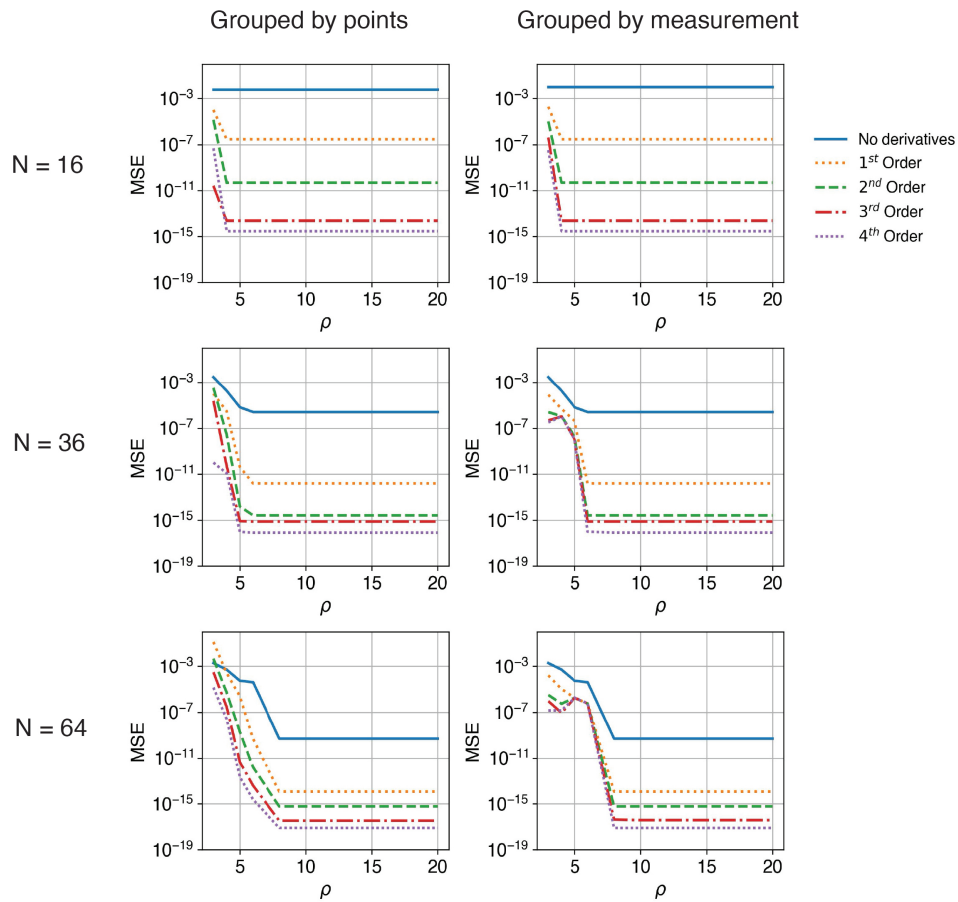


Figure A.1.2: Results of numerical experiments 2D Griewank function from sparse GP for different ρ values and order of derivatives. Figures in the left and right columns show the results of the sparse matrix when the derivatives are grouped by point-wise ordering algorithm 1 and measurement-wise ordering algorithm 1, respectively.

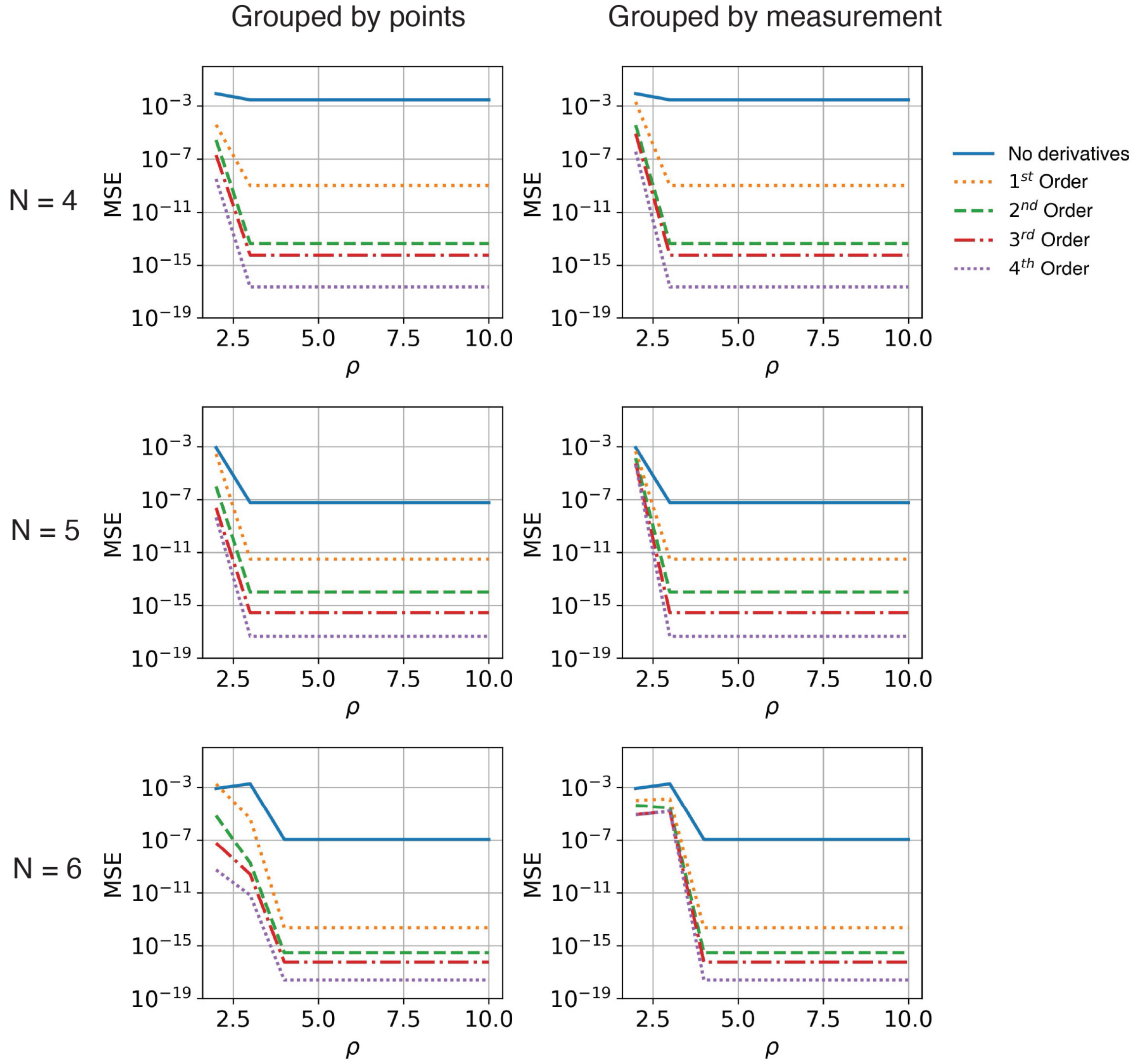


Figure A.1.3: Results of numerical experiments 1D Griewank function from sparse GP for different ρ values and order of derivatives.

For $\rho < \rho_s$, the accuracy of the model is affected by the type of grouping used to include derivative measurements utilized in building the sparse matrix. When the derivatives are grouped by measurement-wise ordering algorithm 1, the prediction error of the model is higher compared to the model when derivatives are grouped by point-wise ordering algorithm 1. For example, Fig.A.1.4 shows the prediction error from measurement grouped by points and measurements grouped by points for $N = 36$ and $\rho = 5$. Note that when the derivatives are grouped by the point-wise ordering algorithm 1, the prediction error reduces with an increase in the order of derivatives; however, the reduction in error is smaller when the derivatives are grouped by the measurement-wise ordering algorithm 1. Similar observations can be made for other N and ρ values. Note that the trend of

the error is unclear when the matrix is really sparse, for example, $\rho = 3$. This suggests that there exists a lower bound of ρ_L below which adding derivatives does not guarantee an improvement in accuracy.

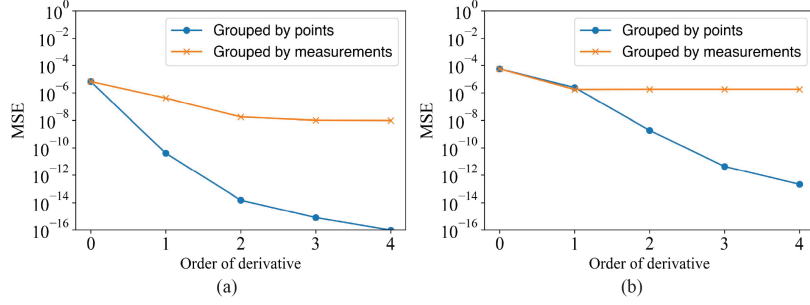


Figure A.1.4: Predictive error of sparse GP for point-wise ordering algorithm 1 and measurement-wise ordering algorithm 1 (a) $N = 36$ and $\rho = 5$, and (b) $N = 64$ and $\rho = 5$.

Now, we will turn our attention to two more ordering algorithms. We call these algorithms “point-wise ordering algorithm 2” and “measurement-wise ordering algorithm 2”. In point-wise ordering algorithm 2, we obtain the ordering \mathbf{P}_{po-2}^d by including derivative measurements using Eq.6. \mathcal{SN}_{po-2}^d are obtained by the procedure described in Section 3.1 using \mathbf{P}_{po-2}^d . Additionally, in the measurement-wise ordering algorithm 2, we obtain \mathbf{P}_{me-2}^d by including derivative measurements at the end of \mathbf{P}_{me-1}^d . \mathcal{SN}_{me-2}^d can then be obtained using \mathbf{P}_{me-2}^d by the procedure described in Section 3.1. Both \mathcal{SN}_{po-2}^d and \mathcal{SN}_{me-2}^d are lists of multiple supernodes used to build the sparse matrix.

Fig.A.1.5 shows a detailed comparison between the four algorithms. The figure compares the predictive performances of point-wise ordering algorithm 1, measurement-wise ordering algorithm 1, point-wise ordering algorithm 2, and measurement-wise ordering algorithm 2 with respect to the order of derivatives for varying numbers of training points, N , and varying ρ values.

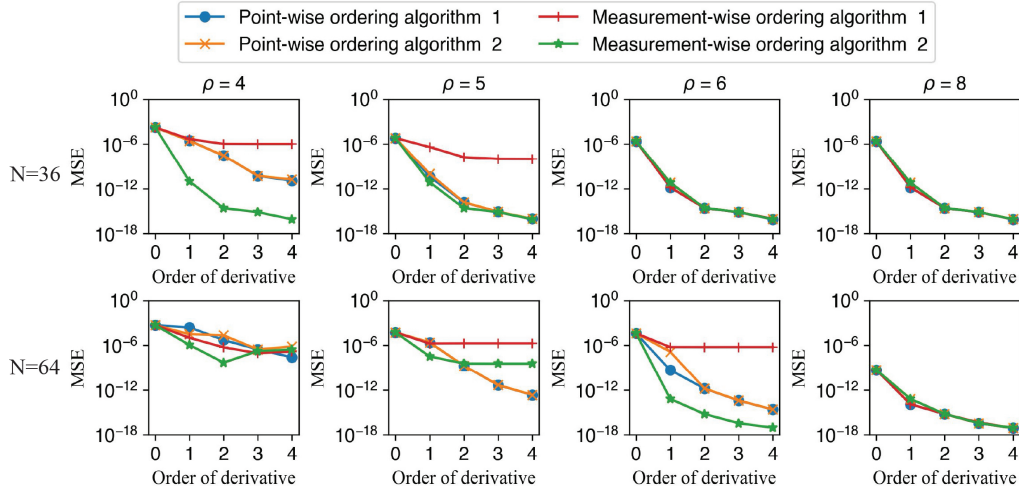


Figure A.1.5: MSE comparison for the four different methods

A.2. Additional Results

This appendix entails additional experimental results performed by varying ρ values and the number of training points. Fig.A.2.1 shows the results of the experiments performed by varying ρ for the 3D Griewank function from the sparse GP. The experiments are performed for point-wise ordering algorithm 1 and measurement-wise ordering algorithm 1 for a fixed number of training points, N , with varying ρ . As the ρ values and the order of derivatives increase, the prediction error reduces for a varying number of training points, N .

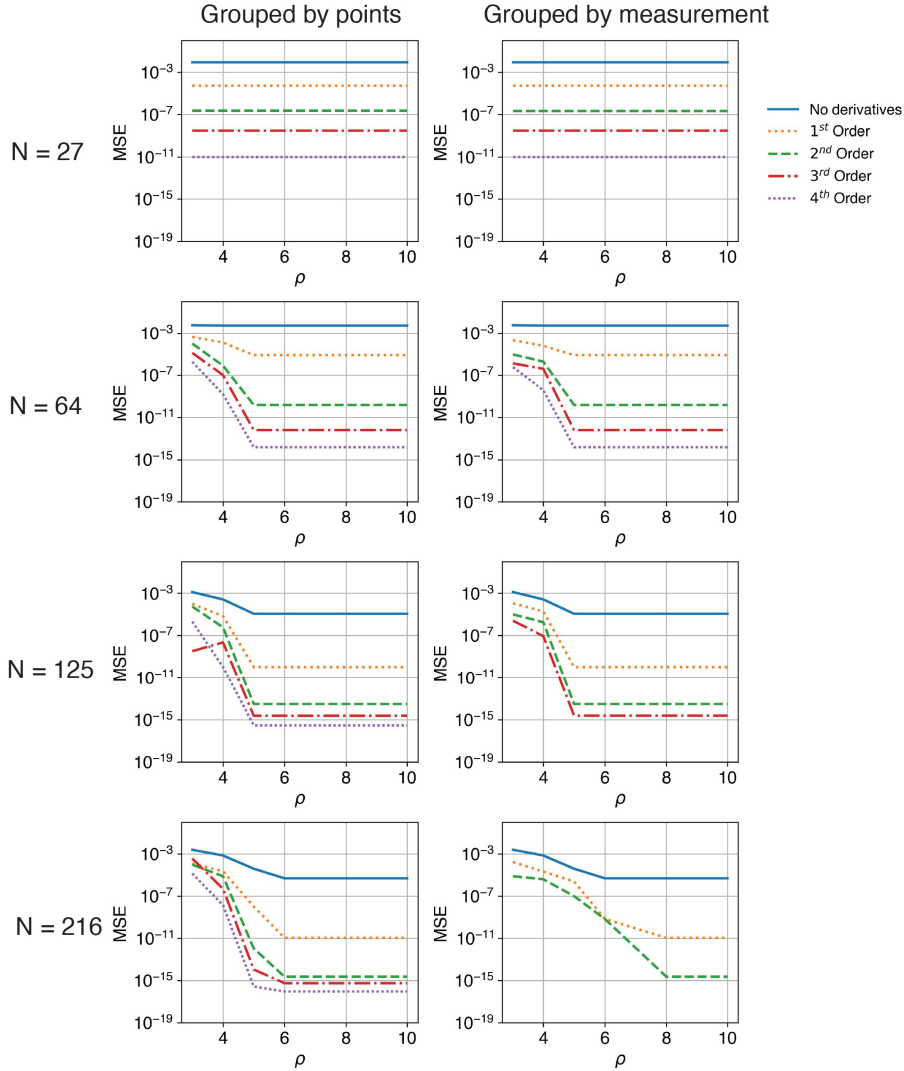


Figure A.2.1: Results of numerical experiments 3D Griewank function from sparse GP for different ρ values and order of derivatives

Fig.A.2.2 shows the results of the experiments performed by varying the number of training points, N , for the 2D Griewank function from the sparse GP. The experiments are performed for point-wise ordering algorithm 1 and measurement-wise ordering algorithm 1 for a fixed ρ value and varying number of training points. As the number of training points and the order of derivatives increase, the prediction error increases until a threshold of $\rho = 8$ is attained, after which the predictive error shows a decreasing trend. In order for the prediction error to decrease with an increase in the number of training points, ρ needs to satisfy the lower bound, $\rho \gtrsim \log\left(\frac{N}{\epsilon}\right)$, mentioned in [43]. In Fig.A.2.2, since we increase the number of training points while keeping ρ fixed, we get an increase in predictive error until the minimum threshold of $\rho = 8$ is reached.

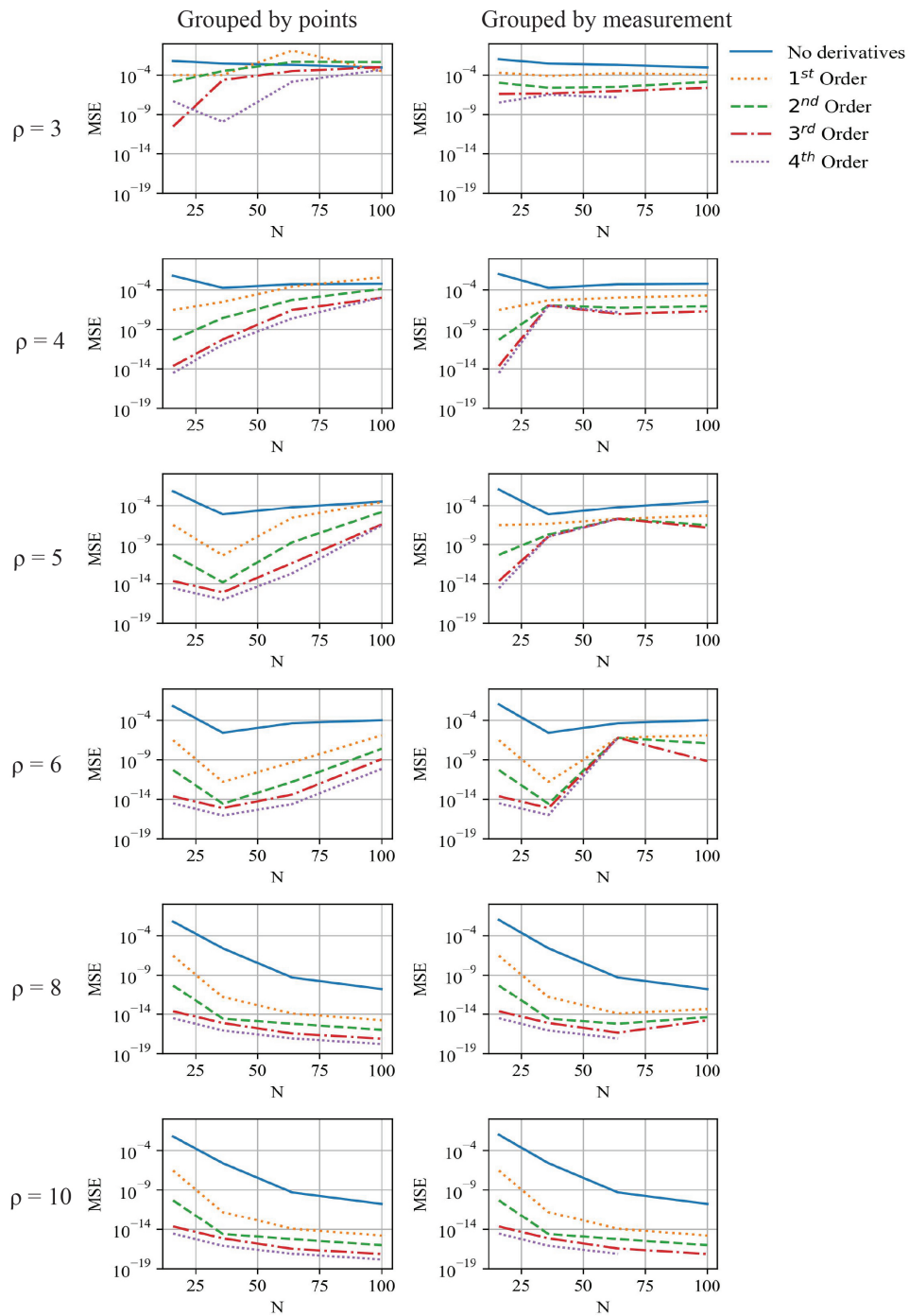


Figure A.2.2: Results of numerical experiments 2D Griewank function from sparse GP for different N , ρ , and order of derivatives

APPENDIX B. ADDITIONAL READINGS

B.1. Details of the Outlier Detection Algorithm Mentioned in Section 4

Algorithm B1 Outlier detection

```

1: procedure IsOUTLIER( $x_{new}, X_{train}$ )
2:    $X_{exist} \leftarrow X_{train}$ 
3:   Let  $D_k$  be an empty list and  $\eta_{out}$  be the percentile set for outlier detection
4:   Let
5:   for each point  $\mathbf{x}_i \in X_{exist}$  do
6:     Let  $d_k(\mathbf{x}_i, X_{exist})$  be the distance from  $\mathbf{x}_i$  to its  $k$ -th nearest neighbor in  $X_{exist}$ .
7:     Append  $d_k(\mathbf{x}_i, X_{exist})$  to  $D_k$ .
8:   end for
9:    $\tau \leftarrow \text{Percentile}(D_k, \eta_{out})$  ▷ The outlier threshold

10:  Let  $d_k(\mathbf{x}_{new}, X_{exist})$  be the distance from  $\mathbf{x}_{new}$  to its  $k$ -th nearest neighbor in  $X_{exist}$ .
11:  if  $d_k(\mathbf{x}_{new}, X_{exist}) > \tau$  then
12:    outlier  $\leftarrow$  True
13:  else
14:    outlier  $\leftarrow$  False
15:  end if
16:  return outlier
17: end procedure

```

B.2. Details of the Decomposition of Σ Mentioned in Section 3

Let \mathbf{K} be the original covariance matrix of the GP without any derivative information, and let \mathbf{R} be a diagonal heteroscedastic noise matrix, where the diagonal elements of \mathbf{R} are given by $\sigma_{n,i}^2 \mathbf{I}$. Then the noisy covariance matrix is given by $\Sigma = \mathbf{K} + \mathbf{R}$. As mentioned in [44], Σ is decomposed into $\Sigma \approx (\mathbf{L}\mathbf{L}^\top)^{-1} \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top \mathbf{R}$ and the incomplete Cholesky factorization with zero fill-in is then applied. The steps involved in going from $\Sigma = \mathbf{K} + \mathbf{R}$ to $\Sigma \approx (\mathbf{L}\mathbf{L}^\top)^{-1} \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top \mathbf{R}$ are provided in this appendix because the authors of [44, 46] did not explicitly discuss the proof in detail.

The precision matrix, \mathbf{K}^{-1} , can be approximately decomposed into $\mathbf{K}^{-1} \approx \mathbf{L}\mathbf{L}^\top$. Since

$$\Sigma = \mathbf{K} + \mathbf{R}$$

we get

$$\Sigma^{-1} = (\mathbf{K} + \mathbf{R})^{-1}$$

Now, using the matrix inversion lemma,

$$\Sigma^{-1} = \mathbf{K}^{-1} - \mathbf{K}^{-1}(\mathbf{I} + \mathbf{R}\mathbf{K}^{-1})^{-1}\mathbf{R}\mathbf{K}^{-1}$$

Or, symmetrically,

$$\Sigma^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}(\mathbf{I} + \mathbf{K}\mathbf{R}^{-1})^{-1}\mathbf{K}\mathbf{R}^{-1} \tag{B.2.1}$$

Note that \mathbf{R} is invertible and cannot be a zero matrix in Eq. (B.2.1). Let, $\mathbf{S} = (\mathbf{K}^{-1} + \mathbf{R}^{-1})^{-1}$, then $(\mathbf{K}^{-1} + \mathbf{R}^{-1})\mathbf{S} = \mathbf{I}$.

Multiplying both sides on the left by \mathbf{K} gives

$$\mathbf{K}(\mathbf{K}^{-1} + \mathbf{R}^{-1})\mathbf{S} = \mathbf{K}$$

which can be simplified by multiplying the \mathbf{K} outside the bracket on the left-hand side of the equation as

$$(\mathbf{I} + \mathbf{K}\mathbf{R}^{-1})\mathbf{S} = \mathbf{K}$$

So, $\mathbf{S} = (\mathbf{I} + \mathbf{K}\mathbf{R}^{-1})^{-1}\mathbf{K}$. Multiplying both sides by \mathbf{R}^{-1} on the right gives

$$\mathbf{S}\mathbf{R}^{-1} = (\mathbf{I} + \mathbf{K}\mathbf{R}^{-1})^{-1}\mathbf{K}\mathbf{R}^{-1} \quad (\text{B.2.2})$$

Substituting the value of $(\mathbf{I} + \mathbf{K}\mathbf{R}^{-1})^{-1}\mathbf{K}\mathbf{R}^{-1}$ from Eq. (B.2.2) into Eq. (B.2.1) gives

$$\boldsymbol{\Sigma}^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{S}\mathbf{R}^{-1}$$

Substituting back $\mathbf{S} = (\mathbf{K}^{-1} + \mathbf{R}^{-1})^{-1}$,

$$\boldsymbol{\Sigma}^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}(\mathbf{K}^{-1} + \mathbf{R}^{-1})^{-1}\mathbf{R}^{-1}$$

Now, given a vector \mathbf{b} , and the equation $(\mathbf{K} + \mathbf{R})\mathbf{y} = \mathbf{b}$, where \mathbf{y} is unknown, we can proceed by multiplying both sides by \mathbf{R}^{-1} on the left

$$(\mathbf{R}^{-1}\mathbf{K} + \mathbf{I})\mathbf{y} = \mathbf{R}^{-1}\mathbf{b}$$

Now, multiplying both sides by \mathbf{K}^{-1} on the left

$$(\mathbf{K}^{-1}\mathbf{R}^{-1}\mathbf{K} + \mathbf{K}^{-1})\mathbf{y} = \mathbf{K}^{-1}\mathbf{R}^{-1}\mathbf{b} \quad (\text{B.2.3})$$

Since, $\mathbf{R}\mathbf{K} \approx \mathbf{K}\mathbf{R}$, then $\mathbf{R}^{-1}\mathbf{K}^{-1} \approx \mathbf{K}^{-1}\mathbf{R}^{-1}$ (for this case of heteroscedastic noise), Eq. (B.2.3) can be simplified as

$$(\mathbf{K}^{-1} + \mathbf{R}^{-1})\mathbf{y} \approx \mathbf{K}^{-1}\mathbf{R}^{-1}\mathbf{b} \quad (\text{B.2.4})$$

Note that we have used $\mathbf{R}\mathbf{K} \approx \mathbf{K}\mathbf{R}$ in the case when \mathbf{R} is the diagonal heteroscedastic noise matrix. In the case where \mathbf{R} is the diagonal homoscedastic noise matrix, the equality holds, i.e., $\mathbf{R}\mathbf{K} = \mathbf{K}\mathbf{R}$.

Substituting $\mathbf{S}^{-1} = (\mathbf{K}^{-1} + \mathbf{R}^{-1})$ in Eq. (B.2.4) gives

$$\mathbf{S}^{-1}\mathbf{y} \approx \mathbf{K}^{-1}\mathbf{R}^{-1}\mathbf{b}$$

which means that,

$$\mathbf{y} \approx \mathbf{S}\mathbf{K}^{-1}\mathbf{R}^{-1}\mathbf{b}$$

Since, $\mathbf{y} = (\mathbf{K} + \mathbf{R})^{-1}\mathbf{b}$, we get

$$(\mathbf{K} + \mathbf{R})^{-1}\mathbf{b} \approx \mathbf{S}\mathbf{K}^{-1}\mathbf{R}^{-1}\mathbf{b}$$

Simplifying,

$$(\mathbf{K} + \mathbf{R})^{-1} \approx \mathbf{S}\mathbf{K}^{-1}\mathbf{R}^{-1} \quad (\text{B.2.4})$$

Since, $\mathbf{S}^{-1} = \mathbf{K}^{-1} + \mathbf{R}^{-1}$ and because we can write $\mathbf{K}^{-1} + \mathbf{R}^{-1} \approx \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top$, we get $\mathbf{S} = (\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top)^{-1}$. Substituting this value of \mathbf{S} into Eq. (B.2.4), we obtain

$$(\mathbf{K} + \mathbf{R})^{-1} \approx (\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top)^{-1}\mathbf{K}^{-1}\mathbf{R}^{-1}$$

which means that

$$\Sigma^{-1} \approx (\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top)^{-1}\mathbf{K}^{-1}\mathbf{R}^{-1}$$

Since $\mathbf{K}^{-1} \approx \mathbf{L}\mathbf{L}^\top$, the above equation can be simplified to

$$\Sigma^{-1} \approx (\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top)^{-1}\mathbf{L}\mathbf{L}^\top\mathbf{R}^{-1}$$

Taking the inverse on both sides, approximating $(\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top)^{-1} \approx \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top$, and approximately letting \mathbf{R} commute gives

$$\Sigma \approx (\mathbf{L}\mathbf{L}^\top)^{-1}\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top\mathbf{R}.$$

□

B.3. Details of the Decomposition of Σ_{der} Mentioned in Section 3

Σ_{der} is given by $\Sigma_{der} = \mathbf{K}_{der} + \mathbf{R}$, where \mathbf{R} is a block diagonal matrix to represent noise. Since the noise term attenuates the exponential decay, we will decompose Σ_{der} . To decompose Σ_{der} , we will rewrite Σ_{der} as

$$\Sigma_{der} = \mathbf{R}^{1/2}(\mathbf{R}^{-1/2}\mathbf{K}_{der}\mathbf{R}^{-1/2} + \mathbf{I})\mathbf{R}^{1/2} \quad (\text{B.3.1})$$

Now we define the whitening transformation as

$$\tilde{\mathbf{K}}_{der} = \mathbf{R}^{-1/2}\mathbf{K}_{der}\mathbf{R}^{-1/2} \quad (\text{B.3.2})$$

Substituting Eq. (B.3.2) into Eq. (B.3.1), we get

$$\Sigma_{der} = \mathbf{R}^{1/2}(\tilde{\mathbf{K}}_{der} + \mathbf{I})\mathbf{R}^{1/2}$$

Taking the inverse on both sides,

$$\Sigma_{der}^{-1} = \mathbf{R}^{-1/2}(\tilde{\mathbf{K}}_{der} + \mathbf{I})^{-1}\mathbf{R}^{-1/2} \quad (\text{B.3.3})$$

Notice that $(\tilde{\mathbf{K}}_{der} + \mathbf{I})^{-1}$ can now be approximated as

$$\mathbf{L}''\mathbf{L}''^\top \approx (\tilde{\mathbf{K}}_{der} + \mathbf{I})^{-1} \quad (\text{B.3.4})$$

because adding \mathbf{I} to $\tilde{\mathbf{K}}_{der}$ corresponds to adding a zero-order term to an elliptic operator, which preserves exponential decay and sparsity.

Define $\mathbf{L}' = \mathbf{R}^{-1/2}\mathbf{L}''$, then $\mathbf{L}'\mathbf{L}'^\top = \mathbf{R}^{-1/2}\mathbf{L}''\mathbf{L}''^\top\mathbf{R}^{-1/2}$.

Substituting Eq. (B.3.4) into the above equation gives

$$\mathbf{L}'\mathbf{L}'^\top \approx \mathbf{R}^{-1/2}(\tilde{\mathbf{K}}_{der} + \mathbf{I})^{-1}\mathbf{R}^{-1/2} = \Sigma_{der}^{-1}$$

Thus, $\Sigma_{der}^{-1} \approx \mathbf{L}'\mathbf{L}'^\top$.

Taking the inverse on both sides gives

$$\Sigma_{der} \approx (\mathbf{L}'\mathbf{L}'^\top)^{-1}$$

where $\mathbf{L}' = \mathbf{R}^{-1/2}\mathbf{L}''$, $(\mathbf{L}''\mathbf{L}''^\top) \approx (\tilde{\mathbf{K}}_{der} + \mathbf{I})^{-1}$, and $\tilde{\mathbf{K}}_{der} = \mathbf{R}^{-1/2}\mathbf{K}_{der}\mathbf{R}^{-1/2}$.

APPENDIX C. PROOFS

C.1. Proof of Lemma 2.3.1.

By construction, the entries of \mathbf{K}_{der} are

$$\mathbf{K}_{\text{der}^{i,j}} = k^{(n_i, n_j)}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}),$$

where $k^{(n_i, n_j)}$ is the mixed partial derivative of the kernel, of order n_i in row i and order n_j in column j .

Since k is positive definite and differentiable, all derivative blocks $k^{(n_i, n_j)}$ satisfy

$$\sum_{i,j} c_i c_j k^{(n_i, n_j)}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \geq 0 \quad \forall \mathbf{c} \in \mathbb{R}^N.$$

\mathbf{K}_{der} is symmetric because mixed derivatives commute for smooth kernels:

$$\frac{\partial^{n_i+n_j} k}{\partial(\mathbf{x}^{(i)})^{n_i} \partial(\mathbf{x}^{(j)})^{n_j}} = \frac{\partial^{n_j+n_i} k}{\partial(\mathbf{x}^{(j)})^{n_j} \partial(\mathbf{x}^{(i)})^{n_i}}.$$

Therefore, \mathbf{K}_{der} is symmetric and positive definite. □

C.2. Proof of Lemma 2.3.2.

The GP posterior variance at a test point \mathbf{x}^* is given by

$$\sigma_d^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{K}_{\text{der},*,d} \mathbf{K}_{\text{der},d}^{-1} \mathbf{K}_{\text{der},*,d}^\top.$$

where $\mathbf{K}_{\text{der}}^d$ is the covariance including derivatives up to order d , and $\mathbf{K}_{\text{der},*,d}$ is the covariance between testing and training points. Now, let \mathbf{K}_{d-1} denote the covariance matrix including derivatives up to order $d-1$, and $\mathbf{K}_{\text{der},*,d-1}$ be the corresponding cross-covariance with \mathbf{x}^* . By construction, adding derivatives of order d adds rows and columns to \mathbf{K}_{d-1} to form \mathbf{K}_d . These additional blocks correspond to the covariance between the new derivative observations and all the previous observations.

Thus, \mathbf{K}_d can be written as a block matrix:

$$\mathbf{K}_d = \begin{bmatrix} \mathbf{K}_{d-1} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix},$$

where \mathbf{B} is the covariance between the order d derivatives and the existing observations, and \mathbf{C} is the covariance between the order d derivatives.

Using Lemma 2.3.1, \mathbf{C} is semi-definite. Similarly, the cross-covariance \mathbf{K}_d can be written as

$$\mathbf{K}_{*,d} = \begin{bmatrix} \mathbf{K}_{*,d-1} & \mathbf{D} \end{bmatrix}.$$

where \mathbf{D} is the covariance between the test point and the order d derivatives.

Furthermore, the posterior variance can be written using the Schur complement as follows:

$$\sigma_d^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \begin{bmatrix} \mathbf{K}_{*,d-1} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{d-1} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{K}_{*,d-1} \\ \mathbf{D} \end{bmatrix}. \quad (\text{C.2.1})$$

By the property of Schur complements for a positive semi-definite block \mathbf{C} :

$$\begin{bmatrix} \mathbf{K}_{d-1} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \succeq \mathbf{K}_{d-1},$$

and therefore, the following inequality holds:

$$\mathbf{K}_{*,d} \mathbf{K}_d^{-1} \mathbf{K}_{*,d}^\top \geq \mathbf{K}_{*,d-1} \mathbf{K}_{d-1}^{-1} \mathbf{K}_{*,d-1}^\top. \quad (\text{C2.2})$$

Using Eq. (C2.2) inequality in the posterior variance formula in Eq. (C2.1) leads to the following expression:

$$\sigma_d^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{K}_{*,d} \mathbf{K}_d^{-1} \mathbf{K}_{*,d}^\top \leq k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{K}_{*,d-1} \mathbf{K}_{d-1}^{-1} \mathbf{K}_{*,d-1}^\top = \sigma_{d-1}^2(\mathbf{x}^*). \quad (\text{C2.3})$$

Moreover, the MSE at a test point \mathbf{x}^* equals the posterior variance of the GP plus the variance due to the noise (if any). Here, by ignoring noise for simplicity, and using Eq. (C2.3), the expression of the MSE can be simplified as follows:

$$\text{MSE}(\hat{f}^d) = \mathbb{E}_{\mathbf{x}^*}[\sigma_d^2(\mathbf{x}^*)] \leq \mathbb{E}_{\mathbf{x}^*}[\sigma_{d-1}^2(\mathbf{x}^*)] = \text{MSE}(\hat{f}^{d-1}).$$

i.e., there exists an error bound. □

C.3. Proof of Lemma 2.3.3.

The kernel is given by

$$k(\mathbf{x}, \mathbf{y}) = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\delta^2}\right),$$

By differentiating the kernel, using the chain and product rules, every mixed partial derivative in the derivative-informed kernel can be written in the form

$$\partial_{\mathbf{x}}^\alpha \partial_{\mathbf{y}}^\beta k(\mathbf{x}, \mathbf{y}) = p_{\alpha,\beta}(\mathbf{x} - \mathbf{y}) k(\mathbf{x}, \mathbf{y}),$$

where $p_{\alpha,\beta}$ is a polynomial whose degree depends only on $|\alpha| + |\beta|$. Hence, the inequality

$$|\partial_{\mathbf{x}}^\alpha \partial_{\mathbf{y}}^\beta k(\mathbf{x}, \mathbf{y})| \leq \sup_{z \in \mathbb{R}^p} |p_{\alpha,\beta}(z)| k(\mathbf{x}, \mathbf{y})$$

holds. By setting $\gamma := 1/(2\delta^2)$ in

$$k(\mathbf{x}, \mathbf{y}) = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\delta^2}\right),$$

and by choosing $C_{\alpha,\beta} := \sigma^2 \sup_z |p_{\alpha,\beta}(z)|$, the following bound is obtained

$$|\partial_{\mathbf{x}}^\alpha \partial_{\mathbf{y}}^\beta k(\mathbf{x}, \mathbf{y})| \leq C_{\alpha,\beta} \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2). \quad \square$$

C.4. Proof of Lemma 3.0.1.

The number of unique derivative terms of order d in p dimensions is given by:

$$\binom{p+d-1}{d}.$$

Multiplying by N gives the total size N_d . The symmetry of \mathbf{K}_{der} follows from kernel derivative symmetry, and positive definiteness simply follows from Lemma 2.3.1.

□

C.5. Proof of Lemma 3.0.2.

Adding higher-order derivatives increases the differences between nearby points, which makes \mathbf{K}_{der} more ill-conditioned. Mathematically, the derivative magnitude scales roughly as

$$\frac{\partial^d k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})}{\partial x^d} \sim \delta^{-d} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}),$$

So increasing d increases the condition number and increasing l smoothens the kernel, which reduces the derivative block magnitude.

□

APPENDIX D. ADDITIONAL THEORETICAL RESULTS

Lemma D.1 (Localization of block covariances). *Let there be two fixed training points $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ and let \mathbf{B}_{ij} denote the covariance block coupling any finite collection of derivative components at $\mathbf{x}^{(i)}$ with any finite collection at $\mathbf{x}^{(j)}$, then there exists constants $C, \gamma > 0$ such that*

$$\|\mathbf{B}_{ij}\| \leq C \exp(-\gamma \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2)$$

Essentially, block coupling decays exponentially with the square of the distance.

Proof. Since each entry of \mathbf{B}_{ij} is of the form $\partial_{\mathbf{x}}^\alpha \partial_{\mathbf{y}}^\beta k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$, so Lemma 2.3.3. gives an exponential bound on each entry. The operator norm of the finite block is bounded by a fixed multiple of the maximal absolute entry, so the same exponential decay holds for the block norm. □

Lemma D.2 (Supernode aggregation and computational cost). *Suppose the columns of the covariance matrix, \mathbf{K}_{der} , are aggregated into n supernodes, each of size at most m , such that each supernode interacts with at most $O(m)$ neighbors, then:*

1. *Building or updating the cholesky factorization under the sparsity structure requires $O(nm^2)$ computational work.*
2. *Restructuring a single supernode in a dynamic update costs $O(m^3)$ arithmetic operation and touches $O(m^2)$ entries.*

Thus, dynamic updates involving only one supernode are substantially cheaper than restructuring the entire factorization.

Proof. Each supernode can be treated as a dense block matrix of size at most m . The dense cholesky factorization of a block of size m costs $O(m^3)$ operations. Since there are n such blocks, and each interacts with only $O(m)$ neighbors, the total work for assembling or updating the global factorization scales as $O(nm^2)$. This includes both the factorization of each supernode and the updates to its neighboring blocks. In the case of a dynamic update where only one supernode changes, the update requires recomputing the dense factorization of its blocks, which costs $O(m^3)$ arithmetic operations. The propagation of updates to adjacent blocks needs modifying $O(m^2)$ entries because each neighboring interaction is at most of size $m \times m$. Thus, the dynamic update cost is cubic in m .

Now consider a dynamic update where only one supernode changes. The update requires re-computing the dense factorization of its block, which costs $O(m^3)$ arithmetic. The propagation of updates to adjacent blocks requires modifying $O(m^2)$ entries, since each neighboring interaction is at most of size $m \times m$. Therefore, the dynamic update cost is cubic in m . \square

Lemma D.3 (Number of supernodes in “point-wise ordering algorithm 1” and “measurement-wise ordering algorithm 1”). *Given a p -dimensional space and derivatives up to the order d . Suppose that the number of supernodes created using point-wise ordering algorithm 1 is SN_p and the number of supernodes created using the measurement-wise ordering algorithm 1 is SN_m . Then, SN_p and SN_m are related by the following equation:*

$$SN_m = z SN_p$$

where z is given by

$$z = \sum_{k=0}^d \binom{p+k-1}{k}$$

Proof. For each measurement type in the measurement-wise ordering algorithm 1, the relative ordering of the points within that type is similar to the point-wise ordering algorithm 1. Hence, the supernode partition that applies to the point-wise ordering algorithm 1 is the same in size within each measurement-type block in the measurement-wise ordering algorithm 1. Therefore, each measurement type within the measurement-wise ordering algorithm 1 contributes exactly SN_p supernodes, so the total number of supernodes in measurement-wise ordering algorithm 1 is given by

$$SN_m = z SN_p$$

Here, z is the number of distinct measurement types per point in the measurement-type ordering algorithm 1.

Now, a partial derivative of f can be indexed by a multi-index

$$\alpha = (\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_p) \in \mathbb{N}^p$$

where, $\nabla^\alpha f(\mathbf{x}) = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \dots \partial x_p^{\alpha_p}}$, $|\alpha| := \alpha_1 + \alpha_2 + \dots + \alpha_p$.

Fixing $k \geq 0$ and then the set of all order k -partial derivatives corresponds to the set

$$A_{p,k} = \alpha \in \mathbb{N}^p : |\alpha| = k$$

The cardinality of this set is the number of nonnegative integer solutions to

$$\alpha_1 + \alpha_2 + \alpha_3 + \dots + \alpha_p = k$$

By the stars-and-bars theorem from combinatorics,

$$|A_{p,k}| = \binom{p+k-1}{k}$$

Summing up to the order of derivatives d gives,

$$\sum_{k=0}^d |A_{p,k}| = \sum_{k=0}^d \binom{p+k-1}{k}$$

which is the value of z , i.e.,

$$z = \sum_{k=0}^d \binom{p+k-1}{k}.$$

□