Fast Gaussian process regression for high dimensional functions with derivative information

Aleksei G Sorokin — *Illinois Institute of Technology, Department of Applied Mathematics, United States*
Sandia National Laboratories.

Pieterjan Robbe — *Sandia National Laboratories.*

Fred J Hickernell — *Illinois Institute of Technology, Department of Applied Mathematics, United States*

Abstract

Gaussian process regression (GPR) is the backbone of many methods in probabilistic numerics. Exact GPR on n sampling locations in d dimensions generally requires $\mathcal{O}(n^2)$ storage and costs $\mathcal{O}(n^3 + dn^2)$ to fit assuming kernel evaluation costs $\mathcal{O}(d)$. Using certain pairings of sampling locations and kernels induces nice structure into the Gram matrix and enables accelerated exact GPR. One popular pairing uses Cartesian grids with product kernels to induce Kronecker structure in the Gram matrix. This reduces exact GP fitting costs to $\mathcal{O}(dn^{3/d})$ and storage requirements to $\mathcal{O}(dn^{2/d})$, but quickly becomes intractable when the dimension exceeds a few dozen. Recent work has shown that pairings certain low discrepancy sequences with special kernels enables GPR to scale like $\mathcal{O}(n \log n + nd)$ for fitting costs and $\mathcal{O}(n)$ for storage requirements. We describe an extension of these methods to problems which observe m derivative multi-indices at each of the n low discrepancy sampling locations. By exploiting similar structure across blocks of the Gram matrix, we are able to reduce the GP fitting cost from $\mathcal{O}(n^3m^3 + n^2m^2d)$ to $\mathcal{O}(m^2 n \log n + m^3 n + nm^2 d)$ and reduce the storage requirements from $\mathcal{O}(n^2m^2)$ to $\mathcal{O}(nm^2)$. We explore a number of synthetic benchmarks to illustrate the potential of the proposed approach.

Proceedings of the 1st International Conference on Probabilistic Numerics (ProbNum) 2025,
Sophia Antipolis, France. PMLR Volume 271. Copyright 2025 with the author(s)

1 Introduction

Gaussian process regression (GPR), also known as kernel interpolation or Kriging, is a Bayesian method for modeling functions and quantifying their uncertainty (Williams and Rasmussen, 2006). The main challenge of exact GPR is that fitting requires $\mathcal{O}(n^2)$ storage and costs $\mathcal{O}(n^3+n^2d)$ computations where we will always assume kernel evaluation costs $\mathcal{O}(d)$. Fast GPR methods exist which cost only $\mathcal{O}(n)$ storage and $\mathcal{O}(n \log n + nd)$ computations while still being exact. We generalize such methods to efficiently incorporate derivative information. While naively incorporating m derivative orders at each of the n points would require $\mathcal{O}(n^2m^2)$ storage and cost $\mathcal{O}(m^3n^3 + m^2n^2d)$, our accelerated methods require only $\mathcal{O}(nm^2)$ storage and cost only $\mathcal{O}(m^2 n \log n + m^3 n + nm^2 d)$. These methods require control over the sampling nodes and the use of special kernels.

GPR has found applications across a broad range of scientific domains including, but not limited to:

Bayesian Optimization (Frazier, 2018; Snoek et al., 2012; Wu et al., 2020) to find global minimum of function which are expensive to evaluate. Here the GPR uncertainty guides sequential sampling to trade off exploration and exploitation.

Solving Partial Differential Equations. For PDEs with deterministic coefficients, often the solution values and derivatives at collocation points are optimized to minimize the norm of the kernel interpolant in the corresponding reproducing kernel Hilbert space (RKHS) (Cockayne et al., 2017; Chen et al., 2021, 2024; Battle et al., 2025; Long et al., 2024). For PDEs with random coefficients, the GPR model often maps the coefficients in a Karhunen–Loève type expansion to the PDE solution from an existing solver (Kaarnioja et al., 2022, 2023; Batlle et al., 2024; Sorokin et al., 2024).

Bayesian Cubature (Briol et al., 2019; O'Hagan, 1991; Rasmussen and Ghahramani, 2003; Jagadeeswaran and Hickernell, 2019, 2022; Rathinavel, 2019), where the integrand is assumed to be a Gaussian process and the posterior distribution of the integral has a closed form Gaussian distribution.

Reliability Analysis (Rackwitz, 2001; Renganathan et al., 2022; Dubourg et al., 2013; Bae et al., 2020; Zanette et al., 2018; Sorokin and Rao, 2023), which is closely related to rare event simulation and level set estimation, tries to quantifying the probability that a simulation subject to random conditions will fail. Oftentimes, as in Bayesian optimization, uncertainty in the GPR model guides sequential sampling to select informant uncertainty settings at which to simulate.

Exact GPR on n points typically costs $\mathcal{O}(n^3 + n^2d)$ as one must evaluate the kernel at all pairs of points and then compute both the inverse and log determinant of an $n \times n$ symmetric positive definite (SPD) Gram matrix $\widetilde{\mathsf{K}}$, which generally costs $\mathcal{O}(n^3)$. This also requires $\mathcal{O}(n^2)$ storage for $\widetilde{\mathsf{K}}$. While this paper focuses on exact GPR methods, we recognize that a number of approximate GPR methods have shown strong performance. Some approximate methods are discussed in Section 7 and related to opportunities to enhance the proposed techniques.

Exact GPR may be made more affordable by forcing

$\widetilde{\mathsf{K}}$ to have additional structure which permits reduced storage requirements for $\widetilde{\mathsf{K}}$ and/or reduced costs to evaluate $\widetilde{\mathsf{K}}^{-1}$ and $\log|\widetilde{\mathsf{K}}|$. Such structures are often induced by pairing a nicely structured covariance kernel with a special design of experiments. For example, pairing product kernels to Cartesian grid sampling locations with $\mathcal{O}(n^{1/d})$ points in each dimension yields Kronecker product Gram matrices for which exact GPR requires only $\mathcal{O}(dn^{3/d})$ computations and $\mathcal{O}(dn^{2/d})$ storage (Saatçi, 2012; Gardner et al., 2018b). Additionally, if the covariance kernel is stationary then each matrix in the Kronecker product becomes Toeplitz and computational costs can be reduced to $\mathcal{O}(dn^{1+1/d})$.

Using grid points in higher dimensions can be intractable and is often undesirable. For example, in dimension $d = 47$ using just $n_j = 2$ grid points in each dimension would require $n = 2^{47} \approx 1.4 \times 10^{14}$ function evaluations whose values would take up over a petabyte of storage in 64 bit precision. Wilson et al. (2014) has proposed methods to use partial grid structure and virtual observations, but they require approximate methods to invert $\widetilde{\mathsf{K}}$ and compute its determinant. Another drawback of using grids is their one dimensional projections only have n_j unique points, which is often quite small for large d. Figure 1 shows 64 grid points in $d = 3$ dimensions with each 1 dimensional projection having only 4 unique values.

We focus on a different class of *fast GPR* methods which yield Gram matrices for which fitting, including optimization of kernel hyperparameters, can be done with only $\mathcal{O}(n \log n + nd)$ computations and $\mathcal{O}(n)$ storage. This is achieved by pairing low discrepancy (LD) sequences with shift invariant (SI) or digitally SI (DSI) covariance kernels. Early derivations of these structures can be found in Zeng et al. (2006) and Zeng et al. (2009). Recently, there has been a renewed interest in using these accelerated GPR methods in the contexts of Bayesian cubature (Jagadeeswaran and Hickernell, 2019, 2022; Rathinavel, 2019) and approximation of PDEs with random coefficients (Kaarnioja et al., 2022, 2023; Batlle et al., 2024; Sorokin et al., 2024). New higher order digitally shift invariant kernels were also derived in Sorokin (2025).

Most popular kernels, such as the squared exponential (SE) and Matérn kernels, are not compatible in this framework. Classes of compatible SI/DSI kernels are presented, and our numerical experiments in Section 6 show they can achieve competitive performance compared to the standard SE GPR method for high dimensional or highly oscillatory problems. These fast GPR methods also require control over the design of experiments. The required LD pointsets are space-filling designs which evenly cover the unit cube $[0,1]^d$. Section 7 discusses how using structured kernel interpolation (SKI) (Wilson and Nickisch, 2015; Gardner et al., 2018b) to accommodate unstructured data is a promising avenue of future work.

Recent works have shown that incorporating derivative observations into GPR can improve performance. In (Eriksson et al., 2018), adding gradient information to GPs is shown to enhance Bayesian optimization and surface/terrain reconstructions. Wu et al. show that exploiting both gradient and Hessian information can improve Bayesian optimization and Bayesian cubature. Padidar et al. provides an extension of variational GPs to include gradient information, and (Solak et al., 2002) discuss gradient-informed GPs for modeling dynamical systems.

GPR with m derivative multi-indices observed at each of the n sampling locations typically requires $\mathcal{O}(n^3m^3 + n^2m^2d)$ computations and $\mathcal{O}(m^2n^2)$ storage. For example, observing a d-dimensional function and it's gradient has $m = 1 + d$, and adding Hessian observations would make $m = 1 + d + d(d+1)/2$. Therefore, exact GPR with derivative observations quickly becomes intractable with moderate n or d. The use of product kernels and regular grids does not immediately facilitate accelerated GPR in the presence of derivative information.

We generalize fast GPR using LD sequences and matching SI/DSI kernels to efficiently utilize (possibly higher order) derivative information. Explicitly, we exploit the similarly nice structures across blocks of the Gram matrix to enables fast GPR with derivative observations to require only $\mathcal{O}(m^2 n \log n + m^3 n + m^2 nd)$ computations and $\mathcal{O}(m^2 n)$ storage.

We begin by reviewing standard GPR in Section 2. Next, Section 3 details fast GPR methods pairing LD points and SI/DSI kernels. Standard GPR with derivatives is reviewed in Section 4, and then fast GPR is generalized to accommodate derivatives in Section 5. Section 6 provides numerical experiments showing the opportunities and drawbacks of fast GPR. A brief summary and directions for future work are discussed in Section 7. The appendix contains common definitions, generalized SI/DSI kernel forms, proofs for DSI kernels, and additional numerical experiments.

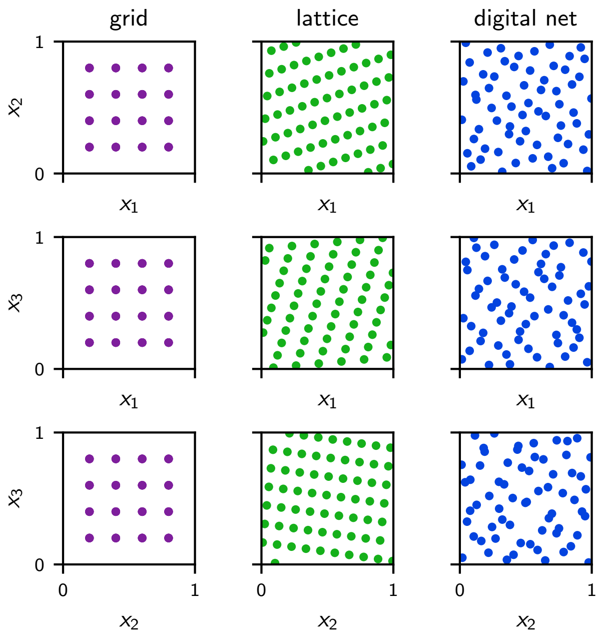

Figure 1: Projections of $n = 64$ points in $d = 3$ dimensions from a regular grid, shifted lattice pointset, and digitally shifted digital net. Notice the lattice and digital net low discrepancy pointsets more evenly fill the unit cube and have more diverse projections.

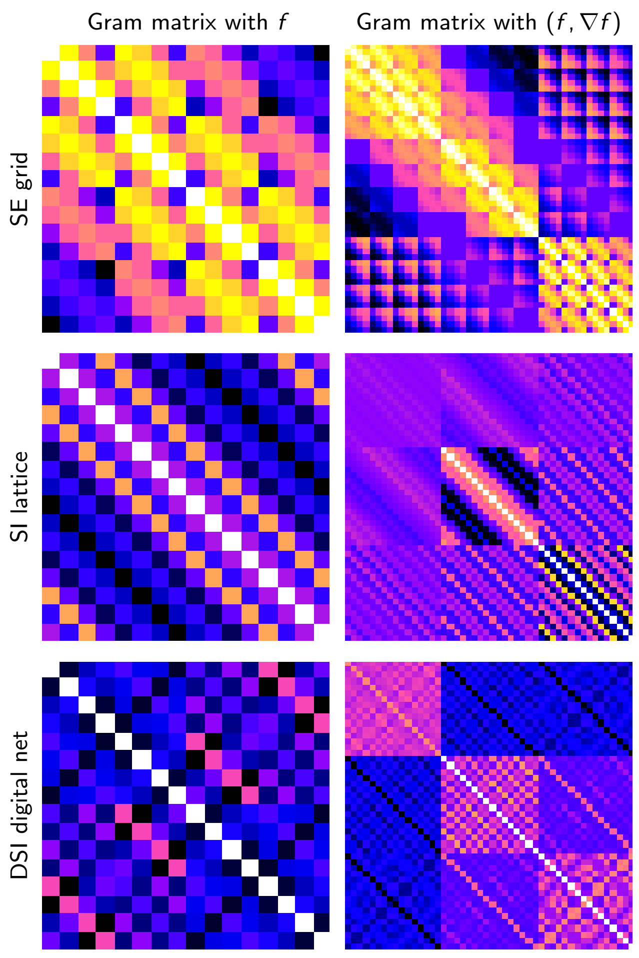

Figure 2: Gram matrix structures when pairing a squared exponential (SE) kernel with regular grid points, a shift invariant (SI) kernel with lattice points, and a digitally SI (DSI) kernel with a digital net. When observing just the function f, the SE kernel with grid points has Kronecker structure, the SI kernel with lattice points has circulant structure, and the DSI kernel with digital net structure has RSBT$_2$ structure. If we include gradient observations, then these same structures can be found in each block of the Gram matrix. Here $n = 16$ points are used in $d = 2$ dimensions.

2 Gaussian process regression

Here we present some background and notation for Gaussian process regression. Greater detail is available in the cornerstone work of (Williams and Rasmussen, 2006).

Let us assume f is a Gaussian process with prior mean M and covariance kernel K which we denote by $f \sim \mathrm{GP}(M, K)$, i.e., $\mathbb{E}[f(\boldsymbol{x})] = M(\boldsymbol{x})$ and $\mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = K(\boldsymbol{x}, \boldsymbol{x}')$. The covariance kernel is required to be symmetric, i.e., $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}', \boldsymbol{x})$, and positive semi-definite, i.e., for any $\boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1}$ and any $c_0, \dots, c_{n-1} \in \mathbb{R}$ we have $\sum_{i,i'=0}^{n-1} c_i c_{i'} K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \geqslant 0$. We will always assume evaluating the kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ costs $\mathcal{O}(d)$.

Suppose we have observed

$$\boldsymbol{y} = \boldsymbol{f}_{\mathsf{X}} + \boldsymbol{\varepsilon} \tag{1}$$

where $\boldsymbol{f}_{\mathsf{X}} = (f(\boldsymbol{x}_i))_{i=0}^{n-1}$ for some d-dimensional sampling locations $\mathsf{X} = (\boldsymbol{x}_i)_{i=0}^{n-1}$. We assume independent noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \xi \mathsf{I})$ where $\xi > 0$ is the common noise variance (nugget) and I is the identity matrix.

The posterior distribution is $f|\boldsymbol{y} \sim \mathrm{GP}(M_n, K_n)$ with posterior mean and covariance given by

$$M_n(\boldsymbol{x}) = M(\boldsymbol{x}) + \boldsymbol{K}_{\mathsf{X}}(\boldsymbol{x})^T \widetilde{\mathsf{K}}^{-1}(\boldsymbol{y} - \boldsymbol{M}_{\mathsf{X}}), \tag{2}$$

$$K_n(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{K}_{\mathsf{X}}(\boldsymbol{x})^T \widetilde{\mathsf{K}}^{-1} \boldsymbol{K}_{\mathsf{X}}(\boldsymbol{x}') \tag{3}$$

respectively. Here $\boldsymbol{K}_{\mathsf{X}}(\boldsymbol{x}) = (K(\boldsymbol{x}, \boldsymbol{x}_i))_{i=0}^{n-1}$ is a vector of kernel evaluations, $\boldsymbol{M}_{\mathsf{X}} = (M(\boldsymbol{x}_i))_{i=0}^{n-1}$ is a vector of prior mean evaluations, and $\widetilde{\mathsf{K}} = \mathsf{K} + \xi \mathsf{I}$ is the noisy version of the Gram matrix $\mathsf{K} = (K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}))_{i,i'=0}^{n-1}$. Notice the posterior covariance only depends on the sampling locations X and kernel K, not the prior mean M. We assume a positive noise variance, $\xi > 0$, so $\widetilde{\mathsf{K}}$ is always symmetric positive definite (SPD).

The sum and product of covariance kernels is itself a valid covariance kernel, see (Duvenaud, 2014, Chapter 2) for a discussion on expressing structure through combining kernels. Here we will consider product kernels

$$K(\boldsymbol{x}, \boldsymbol{x}') = \gamma \prod_{j=1}^{d} R_j(x_j, x_j') \tag{4}$$

where γ is a tuneable global scaling parameter. The squared exponential (SE) kernel, also known as the Gaussian kernel, has

$$R_j(x, x') = \exp\left(-\frac{(x - x')^2}{2\eta_j}\right) \tag{5}$$

where $\boldsymbol{\eta} = (\eta_j)_{j=1}^{d}$ are tuneable lengthscale parameters. We will also consider shift invariant (SI) and digitally SI (DSI) kernels which take the form

$$R_j(x, x') = 1 + \eta_j \mathcal{K}_{\alpha_j}(x \ominus x'). \tag{6}$$

Here $\boldsymbol{\alpha} = (\alpha_j)_{j=1}^{d}$ are fixed smoothness parameters and we defer the definitions of $\mathcal{K}_{\alpha_j}$ and the shift/digital shift $x \ominus x'$ to subsequent subsections. While we only consider product kernels, one may also use more general SI/DSI kernels summing products across subsets of $\{1, \dots, d\}$, see Appendix B for details.

Kernel parameters $\boldsymbol{\theta} := \{\gamma, \boldsymbol{\eta}\}$ are often chosen to maximize the marginal log likelihood (MLL)

$$\mathrm{MLL} = -\frac{(\boldsymbol{y} - \boldsymbol{M}_{\mathsf{X}})^T \widetilde{\mathsf{K}}^{-1}(\boldsymbol{y} - \boldsymbol{M}_{\mathsf{X}}) + \log|\widetilde{\mathsf{K}}| + C}{2}. \tag{7}$$

where $C = n \log(2\pi)$. Thus, GP fitting requires computing coefficients $\widetilde{\mathsf{K}}^{-1}(\boldsymbol{y} - \boldsymbol{M}_{\mathsf{X}})$ and the log determinant $\log|\widetilde{\mathsf{K}}|$. Standard practice is to compute the Cholesky decomposition of $\widetilde{\mathsf{K}}$ which costs $\mathcal{O}(n^3)$ computations and $\mathcal{O}(n^2)$ storage. Inference of the posterior mean (2) and posterior variance (3) can then be performed at $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ cost respectively.

Popular structure-inducing methods set the sampling locations X to lie on a Cartesian grid with $n = \prod_{j=1}^{d} n_j$ points, n_j points in each dimension, and assume K has product form (4). Then $\mathsf{K} = \mathsf{R}_1 \otimes \cdots \otimes \mathsf{R}_d$ where R_j is the $n_j \times n_j$ Gram matrix of R_j evaluated at pairs of grid points in dimension j and $\otimes$ denotes the Kronecker product defined in Definition A.1. For $d > 1$, exploiting this Kronecker structure makes GPR cost $\mathcal{O}(dn^{3/d})$ to fit and require $\mathcal{O}(dn^{2/d})$ storage.

We focus on two alternative pairings of LD pointsets and SI/DSI kernels for which the resulting structure in $\widetilde{\mathsf{K}}$ enables GPR with only $\mathcal{O}(n\log n + nd)$ fitting costs and $\mathcal{O}(n)$ storage. The first pairing uses shifted lattice pointsets and SI kernels to make $\widetilde{\mathsf{K}}$ circulant, i.e., $\widetilde{\mathsf{K}}_{i,i'} = \widetilde{\mathsf{K}}_{0,(i'-i) \mod n}$. The second pairing uses digitally shifted digital nets in base-2 and a DSI kernels to make $\widetilde{\mathsf{K}}$ a 2×2 recursive symmetric block Toeplitz (RSBT$_2$), meaning $\widetilde{\mathsf{K}}$ is a 2×2 symmetric block Toeplitz matrix with square blocks [1], and each of its blocks is a 2×2 symmetric block Toeplitz matrix with square blocks, and each of those blocks .... The key is that circulant and RSBT$_2$ matrices have left eigenvectors $\mathsf{V}^\dagger$ which are proportional to the Fourier and Hadamard matrices respectively. This enables operations with such $\widetilde{\mathsf{K}}$, including computing its inverse, determinant, and eigenvalues, to be done at $\mathcal{O}(n\log n)$ cost using fast Fourier transforms (FFTs) (Cooley and Tukey, 1965) and fast Walsh-Hadamard transforms (FWHTs) (Fino and Algazi, 1976) respectively.

Figure 1 shows a set of grid points, a shifted lattice, and a digitally shifted digital net. Figure 2 shows circulant and RSBT$_2$ structures. When considering derivative information, we will exploit the fact that these same structures appear in each block of the Gram matrix.

3 Fast GPR

Following (Jagadeeswaran and Hickernell, 2019, 2022; Rathinavel, 2019), suppose we can write $\mathsf{K} = \mathsf{V}\Lambda\mathsf{V}^\dagger$ where V is a (possibly complex) symmetric orthonormal matrix with conjugate transpose $\mathsf{V}^\dagger$. Moreover, let us assume the first column of $\mathsf{V}^\dagger$ is $\boldsymbol{v}_1^\dagger = \mathbf{1}/\sqrt{n}$ and that we can compute $\mathsf{V}^\dagger\boldsymbol{a}$ at $\mathcal{O}(n\log n)$ cost for any length n vector $\boldsymbol{a}$.

1. In Section 3.1 we will show that shifted lattice pointsets paired with SI kernels make K circulant. Then $\mathsf{V}^\dagger = \left(e^{-2\pi\sqrt{-1}ii'/n}/\sqrt{n}\right)_{i,i'=0}^{n}$ is the orthonormal Fourier matrix, so $\mathsf{V}^\dagger\boldsymbol{a}$ and $\mathsf{V}\boldsymbol{a}$ can be computed at $\mathcal{O}(n\log n)$ cost using the FFT and inverse FFT (IFFT) of $\boldsymbol{a}$ respectively.
2. In Section 3.2 we will show that digitally shifted base-2 digital nets paired with digitally shift invariant kernels make K RSBT$_2$, so $\mathsf{V}^\dagger$ is the orthonormal Hadamard matrix, see Definition A.2. Then $\mathsf{V}^\dagger = \mathsf{V}$ and $\mathsf{V}^\dagger\boldsymbol{a}$ can be computed using the FWHT at $\mathcal{O}(n\log n)$ cost.

The eigenvalues $\boldsymbol{\lambda} = \mathrm{diag}(\Lambda)$ can also be found at $\mathcal{O}(n\log n)$ cost using the relation

$$\boldsymbol{\lambda} = \sqrt{n}\Lambda\boldsymbol{v}_1^\dagger = \sqrt{n}\mathsf{V}^\dagger\mathsf{V}\Lambda\boldsymbol{v}_1^\dagger = \sqrt{n}\mathsf{V}^\dagger\boldsymbol{k}_1 \tag{8}$$

where $\boldsymbol{k}_1$ is the first column of K and we have used the fact that $\boldsymbol{v}_1^\dagger = \mathbf{1}/\sqrt{n}$. Moreover, $\widetilde{\mathsf{K}} = \mathsf{V}\widetilde{\Lambda}\mathsf{V}^\dagger$ where $\widetilde{\Lambda} = \Lambda + \xi\mathsf{I}$ is the diagonal matrix of noisy eigenvalues. Therefore, we only need $\mathcal{O}(n)$ storage for $\widetilde{\boldsymbol{\lambda}} = \mathrm{diag}(\widetilde{\Lambda})$

[1] M being a 2×2 symmetric block Toeplitz matrix with square blocks means $\mathsf{M} = \begin{pmatrix}\mathsf{A} & \mathsf{B}\\ \mathsf{B} & \mathsf{A}\end{pmatrix}$ for some square blocks A and B.

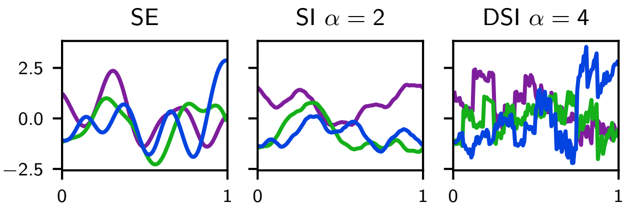

Figure 3: Prior draws from $d = 1$ dimensional GPs having a squared exponential (SE) kernel with $\eta = 10^{-2}$, a shift invariant (SI) kernel with $\alpha = 2$ and $\eta = 1$ (see Section 3.1), and a digitally SI (DSI) kernel with $\alpha = 4$ and $\eta = 1$ (see Section 3.2). The global scaling parameter γ is chosen so all draws have similar ranges. Note the SI draws are periodic and the DSI draws are discontinuous but right differentiable everywhere.

which can be computed at $\mathcal{O}(n\log n + nd)$ cost. The inverse $\widetilde{\mathsf{K}}^{-1} = \mathsf{V}\widetilde{\Lambda}^{-1}\mathsf{V}^\dagger$ and log determinant $\log|\widetilde{\mathsf{K}}| = \log|\widetilde{\Lambda}| = \sum_{i=0}^{n-1}\log|\widetilde{\lambda}_i|$ can also be computed at only $\mathcal{O}(n\log n + nd)$ cost (here we use the fact that all $\widetilde{\boldsymbol{\lambda}} > \mathbf{0}$ since $\widetilde{\mathsf{K}}$ is SPD). Figure 3 plots realizations of GP priors with SE, SI, and DSI kernels.

We say a kernel is (digitally) shift invariant if

$$K(\boldsymbol{x},\boldsymbol{z}) = Q(\boldsymbol{x}\ominus\boldsymbol{z}) \tag{9}$$

for some Q. For the lattice points and shift invariant kernels in Section 3.1, $\boldsymbol{x}\ominus\boldsymbol{z}$ is the elementwise difference modulo 1 as defined in (10). For the digital nets and digitally shift invariant kernels in Section 3.2, $\boldsymbol{x}\ominus\boldsymbol{z}$ is the elementwise exclusive or (XOR) between binary expansions as defined in (13).

We note that for all the (digitally) shift invariant product kernels (4) (6) that we present in the following subsections, we have $\int_{[0,1]^d} K(\boldsymbol{x},\boldsymbol{x}')\mathrm{d}x = \gamma$ for all $\boldsymbol{x}' \in [0,1]^d$. This permits fast Bayesian cubature as discussed in (Rathinavel, 2019).

3.1 Lattices and SI kernels

In this section, for $\boldsymbol{x},\boldsymbol{z}\in[0,1)^d$ we will denote

$$\boldsymbol{x}\ominus\boldsymbol{z} = (\boldsymbol{x}-\boldsymbol{z}) \mod 1. \tag{10}$$

For a fixed generating vector $\boldsymbol{g}\in\{1,\dots,n-1\}^d$ and shift $\boldsymbol{\Delta}\in[0,1]^d$, the shifted lattice pointset is

$$\boldsymbol{x}_i = \frac{i}{n}\boldsymbol{g}\ominus\boldsymbol{\Delta}, \qquad i = 0,\dots,n-1. \tag{11}$$

It is readily shown that $\boldsymbol{x}_i\ominus\boldsymbol{x}_{i'} = \boldsymbol{x}_0\ominus\boldsymbol{x}_{(i'-i) \mod n}$ so that K is circulant. We use a random shift $\boldsymbol{\Delta}\sim\mathcal{U}[0,1]^d$ as commonly done in the literature.

Substantial work has gone into finding good generating vectors $\boldsymbol{g}$ for which the lattice pointset X has low discrepancy with the standard uniform, i.e., the lattice evenly covers the unit cube $[0,1)^d$. We refer interested readers to (Dick et al., 2013, 2022).

One set of product SI kernels (4) (6) have $\mathcal{K}_\alpha(x) = \sum_{k\in\mathbb{Z}\setminus\{0\}}\frac{e^{2\pi\sqrt{-1}kx}}{k^{2\alpha}}$ where $\alpha > 0$ is some smoothness parameter. For $\alpha\in\mathbb{N}$ this simplifies to

$$\mathcal{K}_\alpha(x) = \frac{(-1)^{\alpha+1}(2\pi)^{2\alpha}}{(2\alpha)!}B_{2\alpha}(x) \tag{12}$$

where B_r is the rth Bernoulli polynomial.

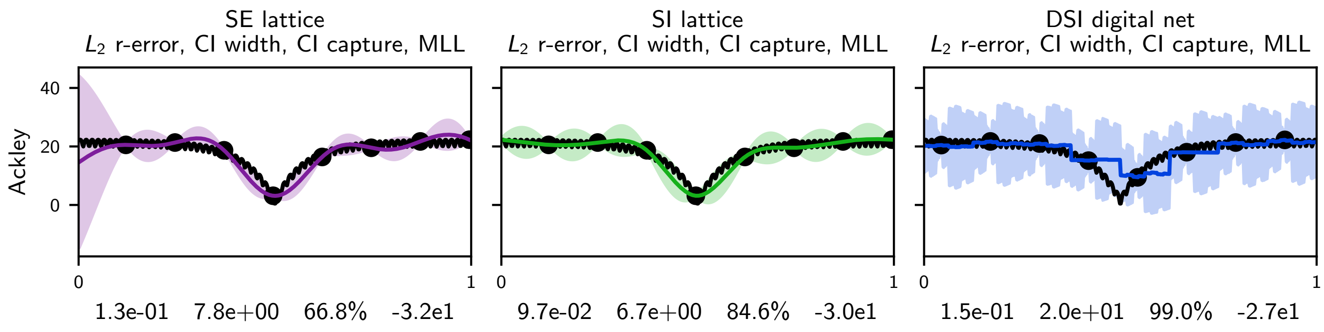

Figure 4: Comparison of GPR methods on the $d = 1$ dimensional Ackley function. Black lines are the true functions, colored lines are the posterior means, and shaded regions show 99% confidence intervals (CIs). The figure lists the L_2 relative errors, the average CI width, the proportion of inputs for which the CI captures the true function, and the optimized MLL.

3.2 Digital nets and DSI kernels

Let $n = 2^p$. Write the base-2 expansion[2] of $0 \leqslant i < n$ as $i = \sum_{\ell=0}^{p-1} 2^\ell \mathsf{i}_\ell$ and $x \in [0,1)$ as $x = \sum_{\ell=1}^\infty \mathsf{x}_\ell 2^{-\ell}$. For $x, z \in [0,1)$ and $\boldsymbol{x}, \boldsymbol{z} \in [0,1)^d$ we write

$$\begin{aligned} x \ominus z &= \sum_{\ell=1}^\infty ((\mathsf{x}_\ell - \mathsf{z}_\ell) \mod 2) 2^{-\ell}, \\ \boldsymbol{x} \ominus \boldsymbol{z} &= (x_1 \ominus z_1, \dots, x_d \ominus z_d) \end{aligned} \tag{13}$$

for the elementwise exclusive or (XOR) between base-2 digits. For fixed $\boldsymbol{x}_{2^0}, \boldsymbol{x}_{2^1}, \dots, \boldsymbol{x}_{2^{p-1}} \in [0,1)^d$ and a digital shift $\boldsymbol{\Delta} \in [0,1]^d$, write the digitally shifted digital net[3] as

$$\boldsymbol{x}_i = \bigominus_{\ell=0}^{p-1} \mathsf{i}_\ell \boldsymbol{x}_{2^\ell} \ominus \boldsymbol{\Delta}, \quad i = 0, \dots, n-1. \tag{14}$$

As was the case with generating vectors for lattices, significant work has gone into finding good generating matrices $\{\boldsymbol{x}_{2^\ell}\}_{\ell=0}^{p-1}$ for which the digital net has low discrepancy. Here we refer the interested reader to (Dick and Pillichshammer, 2010; Dick et al., 2013).[4]

To see that digitally shifted digital nets paired with digitally shift invariant kernels yield RSBT$_2$ K, notice that for $0 \leqslant p' < p$ and $i, i' \in \{0, \dots, 2^{p'-1} - 1\}$ we have

$$\boldsymbol{x}_{i+2^{p'}} \ominus \boldsymbol{x}_{i'} = \boldsymbol{x}_i \ominus \boldsymbol{x}_{2^{p'}} \ominus \boldsymbol{x}_{i'} = \boldsymbol{x}_i \ominus \boldsymbol{x}_{i'+2^{p'}}, \tag{15}$$

$$\boldsymbol{x}_{i+2^{p'}} \ominus \boldsymbol{x}_{i'+2^{p'}} = (\boldsymbol{x}_i \ominus \boldsymbol{x}_{2^{p'}}) \ominus (\boldsymbol{x}_{i'} \ominus \boldsymbol{x}_{2^{p'}}) = \boldsymbol{x}_i \ominus \boldsymbol{x}_{i'}. \tag{16}$$

Using Python-like indexing[5], the $p' = 0$ case shows $\mathsf{K}_{0:2,0:2}$ is SBT$_2$, then the $p' = 1$ case shows $\mathsf{K}_{0:4,0:4}$ is SBT$_2$, and in general the p' case shows $\mathsf{K}_{0:2^{p'+1},0:2^{p'+1}}$ is SBT$_2$.

[2]Here we discuss only base-2 digital nets which permit simpler presentation and more efficient implementation, although natural extensions to prime base $b > 2$ exist (Dick and Pillichshammer, 2010).

[3]Here we technically require the base-2 representation of Δ_j and $x_{2^\ell,j}$ do not end in an infinite trail of ones for any $0 \leqslant \ell < p$ and $1 \leqslant j \leqslant d$. Of course this is not a practical issue in our finite precision implementation.

[4]For readers familiar with Quasi-Monte Carlo, the presented digitally shifted digital nets also accommodate those that have already undergone linear matrix scrambling (LMS), but do not permit nested uniform scrambling (NUS) (Owen, 2003; Sorokin, 2025).

[5]We index columns starting from 0 and do not include the upper index, e.g., $\mathsf{K}_{0:4,0:4}$ contains the first three rows and first three columns of K.

For $k \in \mathbb{N}_0$ write $k = \sum_{\ell=1}^{\#k} 2^{a_\ell - 1}$ where $a_1 > a_2 > \dots > a_{\#k} > 0$ are the indices of the $\#k$ ones in the binary expansion of k, and for integer $\alpha \geqslant 2$ let $\mu_\alpha(k) = \sum_{\ell=1}^{\min(\alpha,\#k)} a_\ell$ sum the α largest indices. Let $\mathrm{wal}_k(x)$ be the k^{th} Walsh coefficient defined as $\mathrm{wal}_k(x) = (-1)^{\sum_{\ell=1}^{\#k} \mathsf{x}_{a_\ell}}$. Then some digitally shift invariant (DSI) product kernels (4) (6) have

$$\mathcal{K}_\alpha(x) = \sum_{k \in \mathbb{N}} \frac{\mathrm{wal}_k(x)}{4^{\mu_\alpha(k)}} \tag{17}$$

where again α controls the smoothness. Closed forms have only been derived for small α, see Sorokin (2025), which draws upon the work of Baldeaux et al. (2012):

$$\begin{aligned} \mathcal{K}_2(x) &= -1 - \beta(x) x + \frac{5}{2}(1 - \tau_1(x)), \\ \mathcal{K}_3(x) &= -1 + \beta(x) x^2 - 5(1 - \tau_1(x)) x \\ &\quad + \frac{43}{18}(1 - \tau_2(x)), \\ \mathcal{K}_4(x) &= -1 - \frac{2}{3}\beta(x) x^3 + 5(1 - \tau_1(x)) x^2 \\ &\quad - \frac{43}{9}(1 - \tau_2(x)) x + \frac{701}{294}(1 - \tau_3(x)) \\ &\quad + \beta(x)\left(\frac{1}{48} S(x) - \frac{1}{42}\right). \end{aligned} \tag{18}$$

Here $\beta(x) = -\lfloor \log_2(x) \rfloor$ is the index of the first one in the binary expansion of x, $\tau_\nu(x) = 2^{-\nu\beta(x)}$, and

$$S(x) = \sum_{\ell \geqslant 0} \frac{(-1)^{\mathsf{x}_{\ell+1}}}{2^{3\ell}}. \tag{19}$$

The $\alpha = 1$ case is considered in (Dick and Pillichshammer, 2005).

Note that the $\mathcal{K}_4$ sum term $S(x)$ can be computed in $\mathcal{O}(\#x)$ time where $\#x$ is the number of ones in the binary expansion of x. An algorithm to compute $\mathcal{K}_\alpha$ for any $\alpha \geqslant 2$ at cost $\mathcal{O}(\alpha \#x)$ is given in (Baldeaux et al., 2012).

4 GPR with derivatives

Here we discuss GPs with derivative information, which may be seen as a special case of optimal recovery in

Hilbert spaces as discussed in (Wendland, 2004, Chapter 16). Let us use the derivative multi-index notations

$$\begin{aligned} f^{(\boldsymbol{\beta})}(\boldsymbol{x}) &:= \partial_{x_1}^{\beta_1}\cdots\partial_{x_d}^{\beta_d} f(\boldsymbol{x}), \\ K^{(\boldsymbol{\beta},\boldsymbol{\kappa})}(\boldsymbol{x},\boldsymbol{z}) &:= \partial_{x_1}^{\beta_1}\cdots\partial_{x_d}^{\beta_d}\partial_{z_1}^{\kappa_1}\cdots\partial_{z_d}^{\kappa_d} K(\boldsymbol{x},\boldsymbol{z}) \end{aligned} \tag{20}$$

where $\boldsymbol{\beta},\boldsymbol{\kappa} \in \mathbb{N}_0^d$. Now, let us assume we observe

$$\boldsymbol{y}^{(\boldsymbol{\beta}_s)} = \boldsymbol{f}_{\mathsf{X}}^{(\boldsymbol{\beta}_s)} + \boldsymbol{\varepsilon}^{(s)}, \qquad s = 1,\dots,m \tag{21}$$

for $\boldsymbol{f}_{\mathsf{X}}^{(\boldsymbol{\beta}_s)} = \left(f^{(\boldsymbol{\beta}_s)}(\boldsymbol{x}_i)\right)_{i=0}^{n-1}$ and derivative multi-indices $\boldsymbol{\beta}_1,\dots,\boldsymbol{\beta}_m \in \mathbb{N}_0^d$. While not strictly justified, we simplify our calculations by assuming that $\boldsymbol{\varepsilon}^{(s)} \sim \mathcal{N}(0,\xi^{(s)}\mathsf{I})$ are all independent. Let us write the m-block vectors $\boldsymbol{y} = \left(\boldsymbol{y}^{(\boldsymbol{\beta}_1)},\dots,\boldsymbol{y}^{(\boldsymbol{\beta}_m)}\right)$ and $\boldsymbol{M}_{\mathsf{X}} = \left(\boldsymbol{M}_{\mathsf{X}}^{(\boldsymbol{\beta}_1)},\dots,\boldsymbol{M}_{\mathsf{X}}^{(\boldsymbol{\beta}_m)}\right)$ where naturally $\boldsymbol{M}_{\mathsf{X}}^{(\boldsymbol{\beta}_s)} = \left(M^{(\boldsymbol{\beta}_s)}(\boldsymbol{x}_i)\right)_{i=0}^{n-1}$. Let $\mathsf{K}^{(\boldsymbol{\beta}_s,\boldsymbol{\beta}_{s'})} = \left(K^{(\boldsymbol{\beta}_s,\boldsymbol{\beta}_{s'})}(\boldsymbol{x}_i,\boldsymbol{x}_{i'})\right)_{i,i'=0}^{n-1}$ be blocks in the $m \times m$ block matrix $\mathsf{K} = \left(\mathsf{K}^{(\boldsymbol{\beta}_s,\boldsymbol{\beta}_{s'})}(\boldsymbol{x}_i,\boldsymbol{x}_{i'})\right)_{s,s'=1}^{m}$. For nuggets (noise variances) $\boldsymbol{\xi} = \left(\xi^{(\boldsymbol{\beta}_s)}\right)_{s=1}^m$ we denote the nugget matrix by $\Xi = \mathrm{diag}(\boldsymbol{\xi})$. Finally we denote by $\widetilde{\mathsf{K}} = \mathsf{K} + \Xi \otimes \mathsf{I}$.

With these notations, we may use our past equations for the posterior mean (2), posterior covariance (3), and MLL (7) (where now $C = mn\log(2\pi)$). For $\boldsymbol{\beta},\boldsymbol{\kappa} \in \mathbb{N}_0^d$, the derivatives of product kernels (4) still take the product form:

$$K^{(\boldsymbol{\beta},\boldsymbol{\kappa})}(\boldsymbol{x},\boldsymbol{z}) = \prod_{j=1}^d R_j^{(\beta_j,\kappa_j)}(x_j,z_j). \tag{22}$$

We note that $K^{(\boldsymbol{\beta},\boldsymbol{\kappa})}$, and derivatives of the (digitally) shift invariant kernels in the following subsections, are not necessarily themselves kernels. Derivatives of the Gaussian kernel (5) can be found using automatic differentiation.

5 Fast GPR with derivatives

Consider j to be fixed and let $\eta := \eta_j$ and $\alpha := \alpha_j$. For $\beta,\kappa \geqslant 0$ the shift invariant kernels (6) satisfy

$$R^{(\beta,\kappa)}(x,z) = 1_{\beta+\kappa=0} + \eta\partial_x^\beta\partial_z^\kappa\mathcal{K}_\alpha(x \ominus z). \tag{23}$$

In the following two subsections we show $\partial_x^\beta\partial_z^\kappa\mathcal{K}_\alpha(x\ominus z)$ is (digitally) shift invariant under certain conditions on $(\alpha,\beta,\kappa,x,z)$.

1. Section 5.1 details the case for shift invariant kernels to be paired with lattices.
2. Section 5.2 details the case for digitally shift invariant kernels to be paired with digital nets, with the understanding that for digitally shift invariant kernels (18) we take right partial derivatives.

To show $\partial_x^\beta\partial_z^\kappa\mathcal{K}_\alpha(x \ominus z)$ is (digitally) shift invariant, we will prove that $\partial_x(x\ominus z), \partial_z(x\ominus z) \in \{-1,1\}$ which implies

$$\partial_x^\beta\partial_z^\kappa\mathcal{K}_\alpha(x\ominus z) \in \left\{\mathcal{K}_\alpha^{(\beta+\kappa)}(x\ominus z), -\mathcal{K}_\alpha^{(\beta+\kappa)}(x\ominus z)\right\}. \tag{24}$$

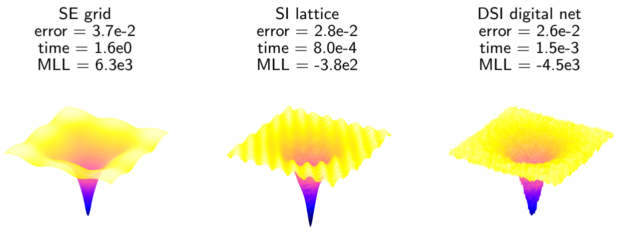

Figure 5: Comparison of GPR posterior mean predictions on the Ackley function in $d = 2$ dimensions sampled at $n = 4096$ points. Metrics for L_2 relative error, time per optimization step, and marginal log likelihood (MLL) are also given. Fig. 7 gives pointwise error plots for these GPs.

Since $\partial_x^\beta\partial_z^\kappa\mathcal{K}_\alpha(x\ominus z)$ retains its (digitally) shift invariant structure, $\mathsf{K}^{(\beta_s,\beta_{s'})} = \mathsf{V}\Lambda^{(\beta_s,\beta_{s'})}\mathsf{V}^\dagger$ for some diagonal matrix $\Lambda^{(\beta_s,\beta_{s'})}$ of eigenvalues $\boldsymbol{\lambda}^{(\beta_s,\beta_{s'})}$. Let $\Lambda = \left(\Lambda^{(\beta_s,\beta_{s'})}\right)_{s,s'=1}^m$ be a $m \times m$ block matrix. Letting $\widetilde{\Lambda} = \Lambda + \Xi \otimes \mathsf{I}$ we have $\widetilde{\mathsf{K}} = (\mathsf{I}\otimes\mathsf{V})\widetilde{\Lambda}(\mathsf{I}\otimes\mathsf{V}^\dagger)$. Since $\widetilde{\Lambda}$ is a $m \times m$ diagonal block matrix, there exists some real permutation matrix P so that $\mathsf{P}^\dagger\widetilde{\Lambda}\mathsf{P} = \widetilde{\Upsilon}$ where $\widetilde{\Upsilon} = \mathrm{diag}\left(\widetilde{\Upsilon}^{[1]},\dots,\widetilde{\Upsilon}^{[n]}\right)$ is a $n \times n$ block diagonal matrix and $\widetilde{\Upsilon}^{[i]}_{s,s'} = \boldsymbol{\lambda}_i^{(\beta_p,\beta_{s'})} + \xi^{(s)}1_{s=s'}$.

In total,

$$\widetilde{\mathsf{K}} = (\mathsf{I}\otimes\mathsf{V})\mathsf{P}\widetilde{\Upsilon}\mathsf{P}^\dagger(\mathsf{I}\otimes\mathsf{V}^\dagger) \tag{25}$$

where $\mathsf{V}^\dagger$ is a fast transform matrix, P is a permutation matrix, and $\widetilde{\Upsilon}$ is a $n\times n$ block diagonal matrix with $m \times m$ blocks of eigenvalues. To represent $\widetilde{\mathsf{K}}$ it is sufficient to know $\widetilde{\Upsilon}$ which requires only $\mathcal{O}(nm^2)$ storage and can be computed at $\mathcal{O}(m^2 n\log n + nm^2 d)$ cost. Thus, the inverse

$$\widetilde{\mathsf{K}}^{-1} = (\mathsf{I}\otimes\mathsf{V})\mathsf{P}\widetilde{\Upsilon}^{-1}\mathsf{P}^\dagger(\mathsf{I}\otimes\mathsf{V}^\dagger) \tag{26}$$

can be computed at $\mathcal{O}(nm^3 + n\log nm^2 + nm^2 d)$ cost since $\widetilde{\Upsilon}^{-1} = \mathrm{diag}\left(\widetilde{\Upsilon}^{-[1]},\dots,\widetilde{\Upsilon}^{-[n]}\right)$. The log determinant

$$\log|\widetilde{\mathsf{K}}| = \log|\Upsilon| = \sum_{i=0}^{n-1}\log\left|\widetilde{\Upsilon}^{[i]}\right| \tag{27}$$

also costs only $\mathcal{O}(nm^3 + n\log nm^2 + nm^2 d)$ compute.

5.1 SI kernels for lattices

For $\boldsymbol{x},\boldsymbol{z} \in [0,1)^d$ let $\boldsymbol{x}\ominus\boldsymbol{z} = (\boldsymbol{x}-\boldsymbol{z}) \mod 1$ as in (10). Clearly $\partial_x(x\ominus z) = 1$ and $\partial_z(x\ominus z) = -1$, so

$$\partial_x^\beta\partial_z^\kappa\mathcal{K}_\alpha(x\ominus z) = (-1)^\kappa\mathcal{K}_\alpha^{(\beta+\kappa)}(x\ominus z). \tag{28}$$

Now, for $\mathcal{K}_\alpha(x)$ defined in (12) and integer $0 \leqslant \chi \leqslant 2(\alpha-1)$, we have $\mathcal{K}_\alpha^{(\chi)}(x) = (2\pi\sqrt{-1})^\chi\mathcal{K}_{\alpha-\chi/2}(x)$. We combine these pieces in the following theorem.

Theorem 5.1. *The derivatives of the shift invariant product kernels take the form of* (22) *with*

$$\begin{aligned} R_j^{(\beta,\kappa)}(x,z) &= 1_{\beta+\kappa=0} \\ &\quad + \eta_j(-1)^\kappa(2\pi\sqrt{-1})^{\beta+\kappa}\mathcal{K}_{\alpha_j-(\beta+\kappa)/2}(x\ominus z) \end{aligned} \tag{29}$$

whenever $\alpha_j \geqslant 2$ and $\beta,\kappa \in \mathbb{N}_0$ satisfy $2\alpha_j - \beta - \kappa \geqslant 2$. When α_j is an integer we have

$$\begin{aligned} R_j^{(\beta,\kappa)}(x,z) &= 1_{\beta+\kappa=0} \\ &\quad + \eta_j\frac{(-1)^{\alpha_j+\kappa+1}(2\pi)^{2\alpha_j}}{(2\alpha_j-\beta-\kappa)!}B_{2\alpha_j-\kappa-\beta}(x) \end{aligned} \tag{30}$$

	L_2 relative error						time per optimization step					
	SE lattice		SI lattice		DSI digital net		SE lattice		SI lattice		DSI digital net	
	f	$(f,\nabla f)$	f	$(f,\nabla f)$	f	$(f,\nabla f)$	f	$(f,\nabla f)$	f	$(f,\nabla f)$	f	$(f,\nabla f)$
Ackley	2.8e-4	3.6e-1	1.0e-4	8.8e-5	1.9e-1	1.9e-2	5.3e-2	2.7e-1	5.7e-4	1.4e-3	1.1e-3	3.0e-3
Branin	6.8e-6	3.5e-7	1.7e-1	2.4e-3	1.8e-2	6.4e-2	5.2e-2	1.0e0	5.2e-4	2.8e-3	1.1e-3	5.9e-3
Camel	2.3e-5	2.5e-6	2.1e-2	1.4e-1	4.3e-2	9.5e-1	4.7e-2	1.0e0	5.5e-4	2.8e-3	1.1e-3	5.8e-3
StyTang	6.6e-5	3.3e-6	3.2e-2	3.5e-1	5.0e-2	4.4e-1	4.9e-2	1.1e0	5.2e-4	2.8e-3	1.1e-3	7.1e-3
Hartmann	3.3e-2	7.8e-3	5.7e-2	3.7e-2	8.0e-2	3.3e-1	8.5e-2	2.2e1	7.5e-4	1.3e-2	1.1e-3	2.6e-2

Table 1: Comparison of GPR error and time with $n = 1024$ points on benchmark functions: Ackley $d = 1$, Branin $d = 2$, Camel $d = 2$, Styblinski–Tang (StyTang) $d = 2$, and Hartman $d = 6$. MLLs are given in Table 2.

where B_r is the r^{th} Bernoulli polynomial.

5.2 DSI kernels for digital nets

Let $\mathbb{B}$ denote the set of all dyadic rationals[6], and let $\Omega(x)$ be the index of the last 1 in that binary expansion of x, e.g., if $x = 3/4 = 0.11_2$ then $\Omega(x) = 2$. Recall $x \ominus z$ is defined in (13) to be the XOR of the binary expansions of x and z. The following lemma and theorem are proved in Appendix C.

Lemma 5.2. *For $x, z \in \mathbb{B}$ the right derivatives satisfy*

$$\partial_x^+(x \ominus z) = \partial_z^+(x \ominus z) \tag{31}$$

Theorem 5.3. *The derivatives of our digitally shift invariant product kernels take the form of (22) with*

$$R_j^{(\beta,\kappa)}(x,z) = 1_{\beta+\kappa=0} + \eta_j(-2)^{\beta+\kappa}(1_{\beta+\kappa>0} + \mathcal{K}_{\alpha_j-\beta-\kappa}(x \ominus z)) \tag{32}$$

for $2 \leqslant \alpha_j \leqslant 4$ and $\beta, \kappa \in \mathbb{N}_0$ satisfying $\alpha_j - \beta - \kappa \geqslant 2$ and $x, y \in \mathbb{B}$ where we understand these to be right derivatives.

6 Numerical Experiments

An implementation of our methods is available in the `fastgps` Python library[7] which builds on Quasi-Monte Carlo tools from `QMCPy`[8] (Choi et al., 2020+, 2022b,a; Sorokin and Rathinavel, 2022; Hickernell et al., 2025; Sorokin, 2025). Our experiments use the $\alpha = 2$ SI kernel (12) and $\alpha = 4$ DSI kernel (18). LD pointsets are generated using `QMCPy`'s default generating vector for lattices and default generating matrices for digital nets. We found our results to be robust to these choices. All optimizations of the MLL were run to convergence using the resilient back propagation (Rprop) algorithm (Riedmiller and Braun, 1993) with learning rate 0.1. Accuracy will be measured by the L_2 relative error

$$\sqrt{\int_{[0,1]^d} |f(x) - M_n(x)|^2 \mathrm{d}x \Big/ \int_{[0,1]^d} |f(x)|^2 \mathrm{d}x} \tag{33}$$

between the benchmark functions f and the posterior means M_n. This is approximated numerically using the ℓ_2 norm and space filling design. All computations were performed on CPUs.

[6]Dyadic rationals are all $x \in [0, 1)$ whose binary expansion has a finite number of ones.
[7]alegresor.github.io/fastgps
[8]qmcsoftware.github.io/QMCSoftware

We are interested in comparing zero-mean GPR with SE (squared exponential), SI, and DSI kernels both with and without gradient information. We note that our benchmarks are smooth, two-sided differentiable functions which do not necessarily align with our fast GP modeling assumptions of either periodicity or only right-differentiability at dyadic rationals. Rather than benchmarking only on functions for which fast GPR is theoretically justified, we are instead interested in seeing how our methods apply to more realistic functions where statistical assumptions may not hold.

We begin in the low-dimensional setting without derivative information. Figure 4 and Fig. 5 show GPR for the $d = 1$ and $d = 2$ dimensional Ackley functions (Ackley, 2012) respectively and we compare across SE GPR with lattice points, fast SI GPR with lattice points, and fast DSI GPR with digital nets. Appendix D includes additional plots in Fig. 8 for a set of visually diverse $d = 1$ curves and a plot for the $d = 2$ dimensional Ackley function in Fig. 7. Notice the periodicity of SE predictions and the discontinuity of DSI predictions as reflected in the kernels.

For the $d = 1$ examples, similar accuracies and MLLs were achieved by all three methods. SE and SI GPR yield similar 99% confidence intervals (CIs)

$$M_n(\boldsymbol{x}) \pm 2.58\sqrt{K_n(\boldsymbol{x},\boldsymbol{x})}. \tag{34}$$

as evidence by their similar (average) CI widths and CI capture proportions. DSI GPR is often less confident in its predictions, but the CI capture proportions always match the desired 99% confidence.

For the $d = 2$ Ackley function fit to $n = 4096$ points, our fast SI and DSI GPR methods achieved slightly better accuracy than the classic SE GPR method and were around 2000 and 1000 times faster respectively. However, the optimized MLL was over 6000 for SE GPR and only around -0.004 for SI and DSI GPR.

Next, Table 1 shows accuracy and timing for GPR using $n = 1024$ points both with and without gradient information on a number of synthetic benchmarks from the Virtual Library of Simulation Experiments (Surjanovic and Bingham). The optimized MLLs are given in Table 2 of Appendix D. For the SI kernel, the periodizing baker transform $\phi(x) = 1 - 2|x - 1/2|$ was used to improve estimates for the gradient-informed Branin and Hartmann functions.

Here we found results to be highly dependent on the benchmark. On the simpler $d = 2$ Branin, Camel, and Styblinski–Tang (StyTang) benchmarks, SE was orders of magnitude more accurate than SI and DSI GPR both

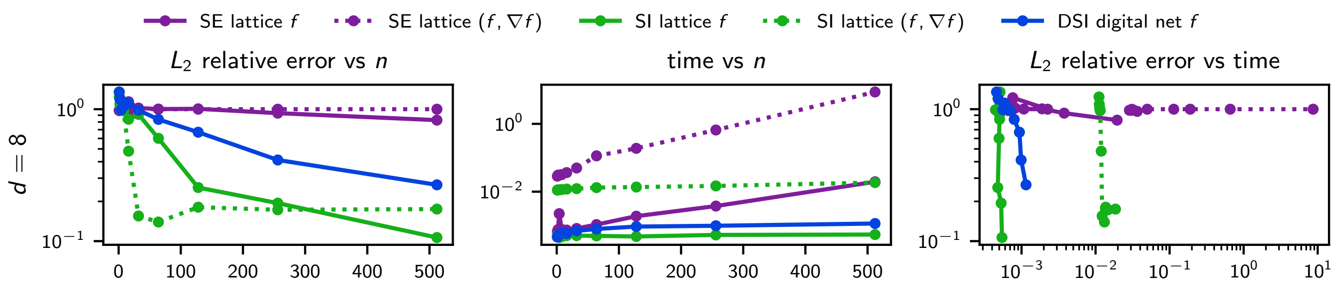

Figure 6: GPR for the $d = 8$ dimensional Styblinski–Tang benchmark. Here we compare the error, number of samples n, time per optimization step, and inclusion of gradient information ∇f. Figure 9 gives more comprehensive results for $d \in \{1, 2, 4, 8, 16\}$.

with and without gradient information. However, on the highly oscillatory $d = 1$ Ackley function, SI is able to achieve comparable accuracy, and both SI and DSI are able to efficiently utilize gradient information while SE cannot. Moreover, on the $d = 6$ Hartmann example, all three methods attain similar accuracy without gradient information. SI was not able to utilize gradient information on either the Camel or StyTang benchmarks, and DSI was only able to utilize gradient information for the Ackley benchmark. This lack of efficiency may be attributed to the benchmarks not satisfying the assumptions of fast GPR as evidenced by the lower MLLs for SI/DSI GPR compared to SE GPR in Table 2.

For GPR on $n = 1024$ points without gradients, SI GPR was consistently around 100 times faster than SE GPR. Including gradients at each of the $n = 1024$ points makes SI over 350 times faster than SE on $d = 2$ examples and almost 1700 times as fast on the $d = 6$ Hartmann function. DSI typically takes around twice as long as SI, partially due to the requirement to compute the $\mathcal{K}_4$ DSI sum term (19). When gradient information is not required, one may use the $\alpha = 2$ and $\alpha = 3$ DSI kernels which are faster to evaluate and did not degrade accuracy in our testing.

Finally, in Fig. 6 we track effects of dimension d and sample size n on the accuracy and time for the Styblinski–Tang benchmark (Styblinski and Tang, 1990). Figure 9 contains a more comprehensive plot comparing across $d \in \{1, 2, 4, 8, 16\}$. We found that as d increases the SI and DSI GPs become significantly more accurate than the SE GPs. For SI kernels and small n, the benefit of gradient information appears to increase with d; this benefit is diminished at a horizon for n that also appears to increase with d. In other words, we found gradient information for fast GPR to be most valuable for large d and small n. DSI with gradients did not show improvement over DSI without gradients and was therefore not included in the plots.

7 Discussion

We proposed novel fast GPR methods which can efficiently incorporate derivative information for functions which are either smooth and periodic (the SI variant) or smooth and right continuous on dyadic rationals (the DSI variant). Empirical testing showed fast SI/DSI GPR can achieve comparable accuracy to standard GPR with the squared exponential (SE) kernel on certain highly oscillatory or high dimensional benchmarks. Moreover, fast SI GPR outperformed standard SE GPR on a Styblinski–Tang benchmark in high dimensions, and gradient information provided significant error reductions for fast SI GPR in high dimensions with low sample sizes. Unfortunately, fast GPR was not able to consistently match standard SE GPR accuracy on all benchmarks, and fast DSI GPR was both conservative in its error estimates and unable to utilize gradient information except in a small number of test cases. Future work should explore a more extensive comparison of our fast GPR methods against other scalable GP methods including, but not limited to, inducing-point variational GPS, structured kernel interpolation (SKI), and preconditioned conjugate gradient (PCG) methods.

Combining these fast SI and DSI methods with approximate GPR may be a promising avenue of future research. Gardner et al. (2018a) proposed to solve the system $\widetilde{\mathsf{K}}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_{\mathsf{X}})$ using PCG methods and to approximate the log determinant $\log \widetilde{\mathsf{K}}$ and its gradient using the Lanczos tridiagonalization algorithm and stochastic trace estimation respectively. For $\widetilde{\mathsf{K}}$ with low condition numbers, PCG converges in $\ll n$ steps and thus has cost $\mathcal{O}(n^2)$. This also enables GPR to exploit fast black box matrix multiplication on GPUs. As the cost of our fast GPR methods is less than that of a single matrix multiplication (at least for small m), we do not see an immediate theoretical advantage to using such methods. Moreover, derivative information is known to give ill conditioned Gram matrices for which PCG would perform poorly or require expensive preconditioners.

Another set of ideas use inducing point methods and structured kernel interpolation (SKI). For an unstructured dataset U of size $\tilde{n}$, standard variational inference (VI) methods are bottlenecked by the requirement to multiply the $n \times \tilde{n}$ kernel matrix K_{UX} with a $\tilde{n} \times \tilde{n}$ Gram matrix K_{UU} which costs $\mathcal{O}(\tilde{n}^2 n)$. Structured kernel interpolation (Wilson and Nickisch, 2015) approximates $\mathsf{K}_{\mathsf{UX}} \approx \mathsf{W}\mathsf{K}_{\mathsf{X},\mathsf{X}}$ for some $\tilde{n} \times n$ weight matrix W which is often sparse. This gives $\mathsf{K}_{\mathsf{UU}} = \mathsf{W}\mathsf{K}\mathsf{W}^T$ where K is nicely structured if using lattice or digital net inducing points. Such methods may be efficiently paired with PCG as multiplication by K_{UU} only requires multiplication by sparse U and nicely structured K at cost $\mathcal{O}(n \log n)$.

Acknowledgment

The authors would like to acknowledge support of NSF grant DMS-231601 and support from the DOE Office of Science Graduate Student Research (SCGSR) Program.

This article has been co-authored by employees of National Technology and Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employees co-own right, title and interest in and to the article and are responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

References

D. Ackley. *A connectionist machine for genetic hillclimbing*, volume 28. Springer science & business media, 2012.

S. Bae, C. Park, and N. H. Kim. Estimating effect of additional sample on uncertainty reduction in reliability analysis using gaussian process. *Journal of Mechanical Design*, 142(11):111706, 2020.

J. Baldeaux, J. Dick, G. Leobacher, D. Nuyens, and F. Pillichshammer. Efficient calculation of the worst-case error and (fast) component-by-component construction of higher order polynomial lattice rules. *Numerical Algorithms*, 59(3):403–431, 2012.

P. Batlle, M. Darcy, B. Hosseini, and H. Owhadi. Kernel methods are competitive for operator learning. *Journal of Computational Physics*, 496:112549, 2024.

P. Batlle, Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart. Error analysis of kernel/gp methods for nonlinear and parametric pdes. *Journal of Computational Physics*, 520:113488, 2025.

F.-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic. Probabilistic integration. *Statistical Science*, 34(1):1–22, 2019.

Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart. Solving and learning nonlinear pdes with gaussian processes. *Journal of Computational Physics*, 447:110668, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110668. URL https://www.sciencedirect.com/science/article/pii/S0021999121005635.

Y. Chen, H. Owhadi, and F. Schäfer. Sparse cholesky factorization for solving nonlinear pdes via gaussian processes. *Mathematics of Computation*, 2024.

S.-C. T. Choi, F. J. Hickernell, R. Jagadeeswaran, M. J. McCourt, and A. G. Sorokin. QMCPy: A Quasi-Monte Carlo Python library, 2020+. URL https://github.com/QMCSoftware/QMCSoftware.

S.-C. T. Choi, Y. Ding, F. J. Hickernell, J. Rathinavel, and A. G. Sorokin. Challenges in developing great Quasi-Monte Carlo software. In *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 209–222. Springer, 2022a.

S.-C. T. Choi, F. J. Hickernell, R. Jagadeeswaran, M. J. McCourt, and A. G. Sorokin. Quasi-Monte Carlo software. In A. Keller, editor, *Monte Carlo and Quasi-Monte Carlo Methods*, pages 23–47, Cham, 2022b. Springer International Publishing. ISBN 978-3-030-98319-2.

J. Cockayne, C. Oates, T. Sullivan, and M. Girolami. Probabilistic numerical methods for pde-constrained bayesian inverse problems. In *AIP Conference Proceedings*, volume 1853. AIP Publishing, 2017.

J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

J. Dick and F. Pillichshammer. Multivariate integration in weighted hilbert spaces based on walsh functions and weighted sobolev spaces. *Journal of Complexity*, 21(2):149–195, 2005.

J. Dick and F. Pillichshammer. *Digital nets and sequences: discrepancy theory and quasi–Monte Carlo integration*. Cambridge University Press, 2010.

J. Dick, F. Kuo, and I. H. Sloan. High dimensional integration — the Quasi-Monte Carlo way. *Acta Numer.*, 22:133–288, 2013. doi: 10.1017/S0962492913000044.

J. Dick, P. Kritzer, and F. Pillichshammer. *Lattice Rules: Numerical Integration, Approximation, and Discrepancy*. Springer Series in Computational Mathematics. Springer Cham, 2022. doi: https://doi.org/10.1007/978-3-031-09951-9.

V. Dubourg, B. Sudret, and F. Deheeger. Metamodel-based importance sampling for structural reliability analysis. *Probabilistic Engineering Mechanics*, 33:47–57, 2013.

D. Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, 2014.

D. Eriksson, K. Dong, E. Lee, D. Bindel, and A. G. Wilson. Scaling gaussian process regression with derivatives. *Advances in neural information processing systems*, 31, 2018.

Fino and Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, 100(11):1142–1146, 1976.

P. I. Frazier. A tutorial on bayesian optimization, 2018. URL https://arxiv.org/abs/1807.02811.

J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018a.

J. Gardner, G. Pleiss, R. Wu, K. Weinberger, and A. Wilson. Product kernel interpolation for scalable gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 1407–1416. PMLR, 2018b.

F. J. Hickernell, N. Kirk, and A. G. Sorokin. Quasi-Monte Carlo methods: What, why, and how? *arXiv preprint arXiv:2502.03644*, 2025.

R. Jagadeeswaran and F. J. Hickernell. Fast automatic bayesian cubature using lattice sampling. *Statistics and Computing*, 29(6):1215–1229, Sep 2019. ISSN 1573-1375. doi: 10.1007/s11222-019-09895-9. URL http://dx.doi.org/10.1007/s11222-019-09895-9.

R. Jagadeeswaran and F. J. Hickernell. Fast automatic bayesian cubature using sobol' sampling. In *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*, pages 301–318. Springer, 2022.

V. Kaarnioja, Y. Kazashi, F. Y. Kuo, F. Nobile, and I. H. Sloan. Fast approximation by periodic kernel-based lattice-point interpolation with application in uncertainty quantification. *Numerische Mathematik*, pages 1–45, 2022.

V. Kaarnioja, F. Y. Kuo, and I. H. Sloan. Lattice-based kernel approximation and serendipitous weights for parametric pdes in very high dimensions. *arXiv preprint arXiv:2303.17755*, 2023.

D. Long, N. Mrvaljević, S. Zhe, and B. Hosseini. A kernel framework for learning differential equations and their solution operators. *Physica D: Nonlinear Phenomena*, 460:134095, 2024.

A. O'Hagan. Bayes–hermite quadrature. *Journal of statistical planning and inference*, 29(3):245–260, 1991.

A. B. Owen. Variance and discrepancy with alternative scramblings. *ACM Transactions of Modeling and Computer Simulation*, 13(4), 2003.

M. Padidar, X. Zhu, L. Huang, J. Gardner, and D. Bindel. Scaling gaussian processes with derivative information using variational inference. *Advances in Neural Information Processing Systems*, 34:6442–6453, 2021.

R. Rackwitz. Reliability analysis—a review and some perspectives. *Structural safety*, 23(4):365–395, 2001.

C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. *Advances in neural information processing systems*, pages 505–512, 2003.

J. Rathinavel. *Fast automatic Bayesian cubature using matching kernels and designs.* Illinois Institute of Technology, 2019.

S. A. Renganathan, V. Rao, and I. M. Navon. Camera: A method for cost-aware, adaptive, multifidelity, efficient reliability analysis, 2022. URL https://arxiv.org/abs/2203.01436.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE international conference on neural networks*, pages 586–591. IEEE, 1993.

Y. Saatçi. *Scalable inference for structured Gaussian process models.* PhD thesis, Citeseer, 2012.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

E. Solak, R. Murray-Smith, W. Leithead, D. Leith, and C. Rasmussen. Derivative observations in gaussian process models of dynamic systems. *Advances in neural information processing systems*, 15, 2002.

A. Sorokin. A unified implementation of quasi-Monte Carlo generators, randomization routines, and fast kernel methods, 2025. URL https://arxiv.org/abs/2502.14256.

A. G. Sorokin and V. Rao. Credible intervals for probability of failure with gaussian processes, 2023.

A. G. Sorokin and J. Rathinavel. On bounding and approximating functions of multiple expectations using quasi-Monte Carlo. In *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 583–599. Springer, 2022.

A. G. Sorokin, A. Pachalieva, D. O'Malley, J. M. Hyman, F. J. Hickernell, and N. W. Hengartner. Computationally efficient and error aware surrogate construction for numerical solutions of subsurface flow through porous media. *Advances in Water Resources*, 193:104836, 2024. ISSN 0309-1708. doi: https://doi.org/10.1016/j.advwatres.2024.104836. URL https://www.sciencedirect.com/science/article/pii/S0309170824002239.

M. Styblinski and T.-S. Tang. Experiments in non-convex optimization: stochastic approximation with function smoothing and simulated annealing. *Neural Networks*, 3(4):467–483, 1990.

S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved February 19, 2025, from http://www.sfu.ca/~ssurjano.

H. Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

A. Wilson and H. Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International conference on machine learning*, pages 1775–1784. PMLR, 2015.

A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cunningham. Fast kernel learning for multidimensional pattern extrapolation. *Advances in neural information processing systems*, 27, 2014.

A. Wu, M. C. Aoi, and J. W. Pillow. Exploiting gradients and hessians in bayesian optimization and bayesian quadrature. *arXiv preprint arXiv:1704.00060*, 2017.

J. Wu, S. Toscano-Palmerin, P. I. Frazier, and A. G. Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. In *Uncertainty in Artificial Intelligence*, pages 788–798. PMLR, 2020.

A. Zanette, J. Zhang, and M. J. Kochenderfer. Robust super-level set estimation using gaussian processes, 2018. URL https://arxiv.org/abs/1811.09977.

X. Zeng, K.-T. Leung, and F. J. Hickernell. *Error analysis of splines for periodic problems using lattice designs.* Springer, 2006.

X. Zeng, P. Kritzer, and F. J. Hickernell. Spline methods using integration lattices and digital nets. *Constructive Approximation*, 30:529–555, 2009.

A Definitions

Definition A.1 (Kronecker Product). *For A an $m \times n$ matrix and B a $p \times q$ matrix, the Kronecker product $\mathsf{A} \otimes \mathsf{B}$ is the $pm \times qn$ block matrix*

$$\mathsf{A} \otimes \mathsf{B} = \begin{pmatrix} a_{11}\mathsf{B} & \cdots & a_{1n}\mathsf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathsf{B} & \cdots & a_{mn}\mathsf{B} \end{pmatrix}.$$

Definition A.2 (Orthonormal Hadamard Matrix). *The Hadamard matrix of size 1 is*

$$H_1 = \begin{pmatrix} 1 \end{pmatrix}$$

and the Hadamard matrix of size 2^p for $p > 0$ is

$$H_{2^p} = \begin{pmatrix} H_{2^{p-1}} & H_{2^{p-1}} \\ H_{2^{p-1}} & -H_{2^{p-1}} \end{pmatrix}.$$

The orthonormal Hadamard matrix is $H_{2^p}/\sqrt{2^p}$.

B General SI/DSI Kernels

In (4) and (6) we present SI/DSI product kernels of the form

$$K(\boldsymbol{x}, \boldsymbol{x}') = \gamma \prod_{j=1}^{d} \left[1 + \eta_j \mathcal{K}_{\alpha_j}(x_j \ominus x_j')\right].$$

Without sacrificing shift invariance or the properties enabling fast GPR, one may instead use the more general kernels of the form

$$K(\boldsymbol{x}, \boldsymbol{x}') = \gamma \sum_{\mathfrak{u} \subseteq \{1,\dots,d\}} \widetilde{\eta}_{\mathfrak{u}} \prod_{j \in \mathfrak{u}} \mathcal{K}_{\alpha_j}(x_j \ominus x_j') \qquad (35)$$

where the $\mathfrak{u} = \varnothing$ term equals 1. Here γ is still a global scaling parameter, but we now use different lengthscales $\widetilde{\eta}_{\mathfrak{u}}$ for each subset of dimensions $\mathfrak{u} \subseteq \{1,\dots,d\}$. The product kernels are a special case where $\widetilde{\eta}_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \eta_j$ for some per-dimension lengthscales $\{\eta_j\}_{j=1}^d$. Note that the full SI/DSI kernel in (35) costs $\mathcal{O}(2^d)$ to evaluate instead of the $\mathcal{O}(d)$ cost assumed throughout the main text.

C Right derivatives of digitally shift invariant kernels

Proof of Lemma 5.2. Let $\Omega = \max\{\Omega(x), \Omega(z)\}$. For $h < 2^{-\Omega}$ we have $(x+h) \ominus z = (x \ominus z) + h$ so the right partial derivative satisfies

$$\partial_x^+(x \ominus z) = \lim_{h \to 0^+} \frac{((x+h) \ominus z) - (x \ominus z)}{h} = 1. \qquad (36)$$

However, for $h < 2^{-\Omega}$ we have $(x-h) \ominus z \geqslant 2^{-a(x)}$ so the left partial derivative is $\partial_x^-(x \ominus z) = \infty$. The same arguments can be used to show $\partial_z^+(x \ominus z) = 1$ while $\partial_z^-(x \ominus z) = \infty$. $\square$

Lemma C.1. *For $x \in \mathbb{B}$ and $S(x)$ defined in (19) the partial derivatives satisfy*

$$\partial^+ S(x) = 0 \qquad \text{and} \qquad \partial^- S(x) = -\infty. \qquad (37)$$

Proof. Let $\Omega = \Omega(x)$ and recall $\beta(x) = -\lfloor \log_2(x) \rfloor$ is the index of the first 1 in the binary expansion of x. For $h < 2^{-\Omega}$ we have

$$S(x+h) - S(x) = \sum_{\ell \geqslant \Omega} \left[\frac{(-1)^{\mathsf{h}_{\ell+1}}}{2^{3\ell}} - \frac{1}{2^{3\ell}}\right]. \qquad (38)$$

Clearly $\partial^+ S(x) \leqslant 0$. However,

$$\begin{aligned} \partial^+ S(x) &\geqslant \lim_{h \to 0^+} \frac{1}{h} \sum_{\ell \geqslant \beta(h)-1} \left[\frac{-1}{2^{3\ell}} - \frac{-1}{2^{3\ell}}\right] \\ &= -\lim_{h \to 0^+} \frac{1}{h} \sum_{\ell \geqslant \beta(h)-1} \frac{1}{2^{3\ell-1}} \\ &= -\frac{2^7}{7} \lim_{h \to 0^+} \frac{2^{-3\beta(h)}}{h} \\ &= 0, \end{aligned}$$

so $\partial^+ S(x) = 0$.

The left derivative, assuming $x \in \mathbb{B}\backslash\{0\}$, is

$$\begin{aligned} \partial^- S(x) &= \lim_{h \to 0^+} \frac{S(x) - S(x-h)}{h} \\ &\leqslant \lim_{h \to 0^+} \frac{1}{h}\Bigg[\left(-\frac{1}{2^{3(\Omega-1)}} + \sum_{\ell \geqslant \Omega} \frac{1}{2^{3\ell}}\right) \\ &\quad - \left(\frac{1}{2^{3(\Omega-1)}} - \sum_{\ell \geqslant \Omega} \frac{1}{2^{3\ell}}\right)\Bigg] \\ &= \lim_{h \to 0^+} \frac{1}{h} \left(\sum_{\ell \geqslant \Omega} \frac{1}{2^{3\ell-1}} - \frac{1}{2^{3\Omega-4}}\right) \\ &= -\frac{3}{7} 2^{5-3\Omega} \left(\lim_{h \to 0^+} \frac{1}{h}\right) \\ &= -\infty. \end{aligned}$$

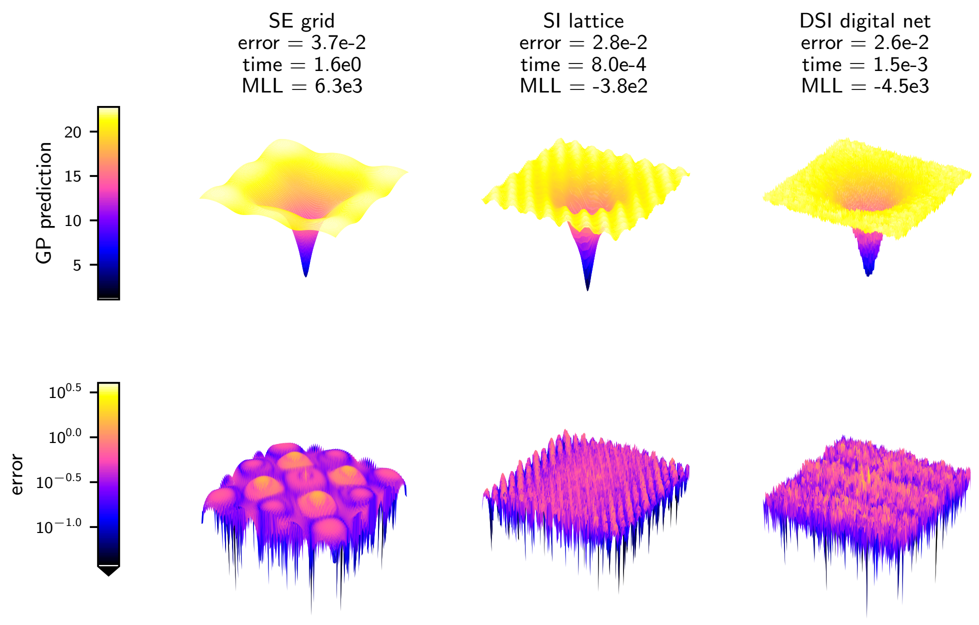

Figure 7: This plot extends Fig. 5 to show posterior mean predictions and pointwise errors for GPs with $n = 4096$ points fit to the $d = 2$ dimensional Ackley function.

	MLL (Marginal Log Likelihood)					
	SE lattice		SI lattice		DSI digital net	
	f	$(f, \nabla f)$	f	$(f, \nabla f)$	f	$(f, \nabla f)$
Ackley	8.7e2	-1.9e7	5.0e2	-4.7e3	-1.8e3	-1.2e4
Branin	3.4e3	1.3e4	-4.8e3	-1.4e4	-2.8e3	-2.1e4
Camel	3.3e3	1.0e4	-1.8e3	-1.4e4	-1.0e3	-2.3e4
StyTang	3.4e3	1.1e4	-2.4e3	-1.8e4	-8.5e2	-2.4e4
Hartmann	1.1e3	6.7e3	6.5e2	-5.5e3	6.2e2	-5.7e3

Table 2: Comparison of GPR MLL on benchmark functions: Ackley $d = 1$, Branin $d = 2$, Camel $d = 2$, Styblinski–Tang (StyTang) $d = 2$, and Hartman $d = 6$. Table 1 gives accuracy and timing metrics.

$\square$

Lemma C.2. *For $x \in \mathbb{B}$ we have*

$$\partial^+ \mathcal{K}_4(x) = -2(1 + \mathcal{K}_3(x)),$$
$$\partial^+ \mathcal{K}_3(x) = -2(1 + \mathcal{K}_2(x)).$$

Proof. Recall $\beta(x) = -\lfloor \log_2(x) \rfloor$ is the index of the first 1 in the binary expansion of x. For any $x \in \mathbb{B} \backslash \{0\}$ we can choose $h > 0$ small enough so that $\beta(x) = \beta(x + h)$. Combined with Lemma C.1, this implies $\mathcal{K}_4$ and $\mathcal{K}_3$ are both right differentiable on $\mathbb{B} \backslash \{0\}$. It may be readily checked that

$$\partial^+ \mathcal{K}_3(0) = \lim_{h \to 0^+} \left[\frac{\beta(h) h^2 - 5\,(1 - \tau_1(h))\,h}{h} + \frac{\frac{43}{18}\,(1 - \tau_2(h)) - \frac{43}{18}}{h} \right] = -5.$$

We also claim that

$$\partial^+ \mathcal{K}_4(0) = \lim_{h \to 0^+} \left[\frac{-\frac{2}{3}\beta(h) h^3 + 5\,(1 - \tau_1(h))\,x^2}{h} + \frac{-\frac{43}{9}\,(1 - \tau_2(h))\,h + \frac{701}{294}\,(1 - \tau_3(h)) - \frac{701}{294}}{h} \right] = -43/9.$$

To see this, note that

$$\lim_{h \to 0^+} \frac{\beta(h)}{h} \left(\frac{1}{48} \sum_{\ell \geqslant 0} \frac{(-1)^{\mathsf{h}_{\ell+1}}}{2^{3\ell}} - \frac{1}{42} \right) = 0. \tag{39}$$

The above equation follows from writing

$$\begin{aligned} &\frac{1}{48} \sum_{\ell \geqslant 0} \frac{(-1)^{\mathsf{h}_{\ell+1}}}{2^{3\ell}} - \frac{1}{42} \\ &= \frac{1}{48} \left(\sum_{\ell < \beta(h) - 1} \frac{1}{2^{3\ell}} - \frac{1}{2^{3(\beta(h)-1)}} + \sum_{\ell > \beta(h) - 1} \frac{(-1)^{\mathsf{h}_{\ell+1}}}{2^{3\ell}} \right) - \frac{1}{42} \\ &\in \left[-\frac{8}{21} 2^{-3\beta(h)}, -\frac{1}{3} 2^{-3\beta(h)} \right] \end{aligned} \tag{40}$$

so that

$$\begin{aligned} 0 &= -\frac{8}{21} \left(\lim_{h \to 0^+} \frac{\beta(h)}{h 2^{3\beta(h)}} \right) \\ &\leqslant \lim_{h \to 0^+} \frac{\beta(h)}{h} \left(\frac{1}{48} \sum_{\ell} \frac{(-1)^{\mathsf{h}_{\ell+1}}}{2^{3\ell}} - \frac{1}{42} \right) \\ &\leqslant -\frac{1}{3} \left(\lim_{h \to 0^+} \frac{\beta(h)}{h 2^{3\beta(h)}} \right) \\ &= 0. \end{aligned} \tag{41}$$

$\square$

Proof of Theorem 5.3. Combining Lemma 5.2 and Lemma C.2 gives

$$\partial_x^+ \mathcal{K}_\alpha(x, y) = -2\,(1 + \mathcal{K}_{\alpha-1}(x, y)) \tag{42}$$

from which the theorem follows. $\square$

D Additional Numerical Experiments

Figure 8 extends Fig. 4 in comparing standard GPR to the presented fast GPR methods, all without derivative information, on a set of visually diverse $d = 1$ curves. Figure 7 extends Figure 5 to include pointwise errors for the $d = 2$ dimensional GPs fit to the Ackley function. Table 2 extends Table 1 to provide optimized MLLs across benchmarks. Figure 9 extends Fig. 6 to provide a more complete set of results for our experiment exploring trade offs in dimension, accuracy, time, and derivative information.

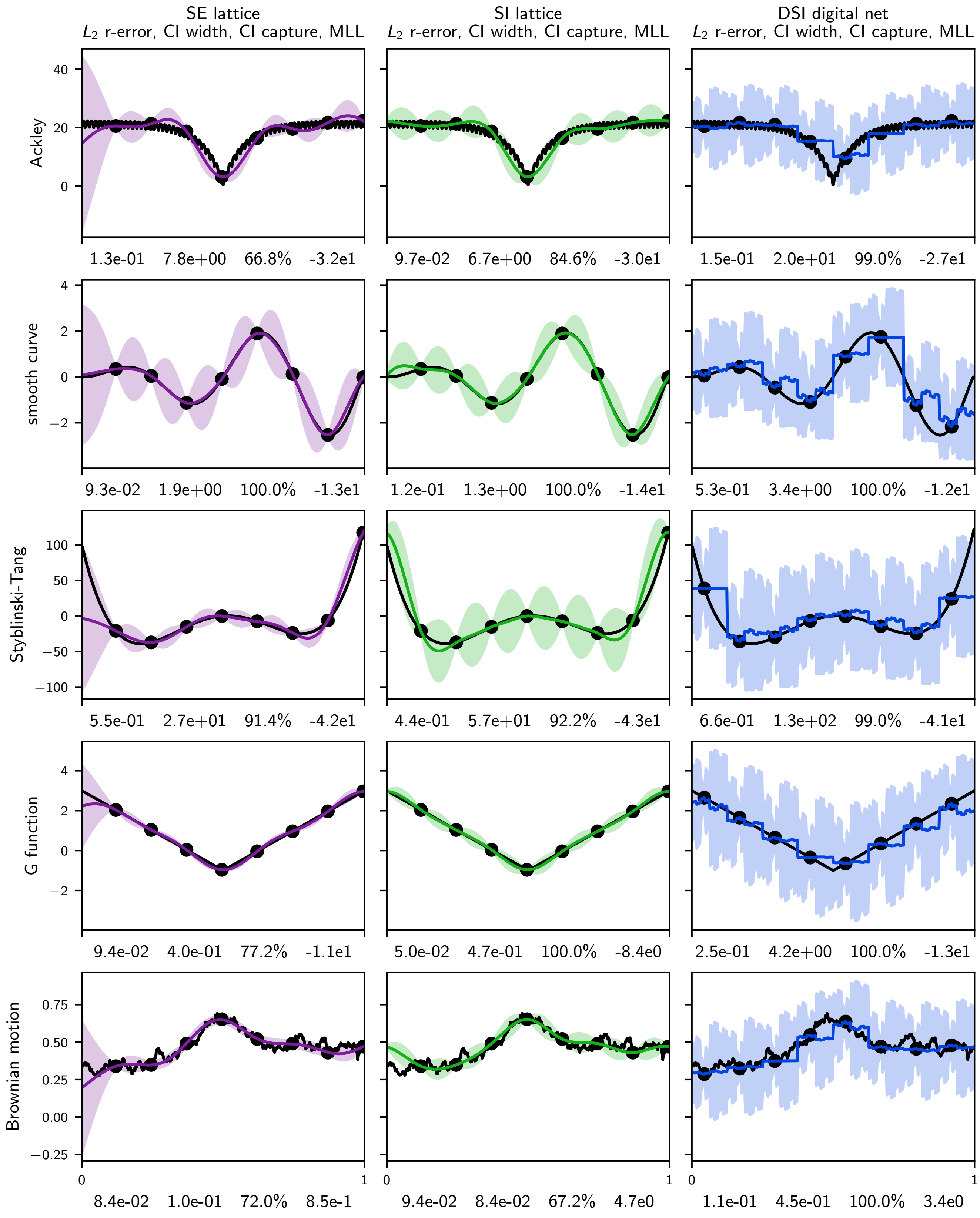

Figure 8: Comparison of GPR methods on visually diverse example functions in dimension $d = 1$. Black lines are the true functions, colored lines are the posterior means, and shaded regions show 99% confidence intervals (CIs). The figure lists the L_2 relative errors, the average CI width, the proportion of inputs for which the CI captures the true function, and the optimized MLL.

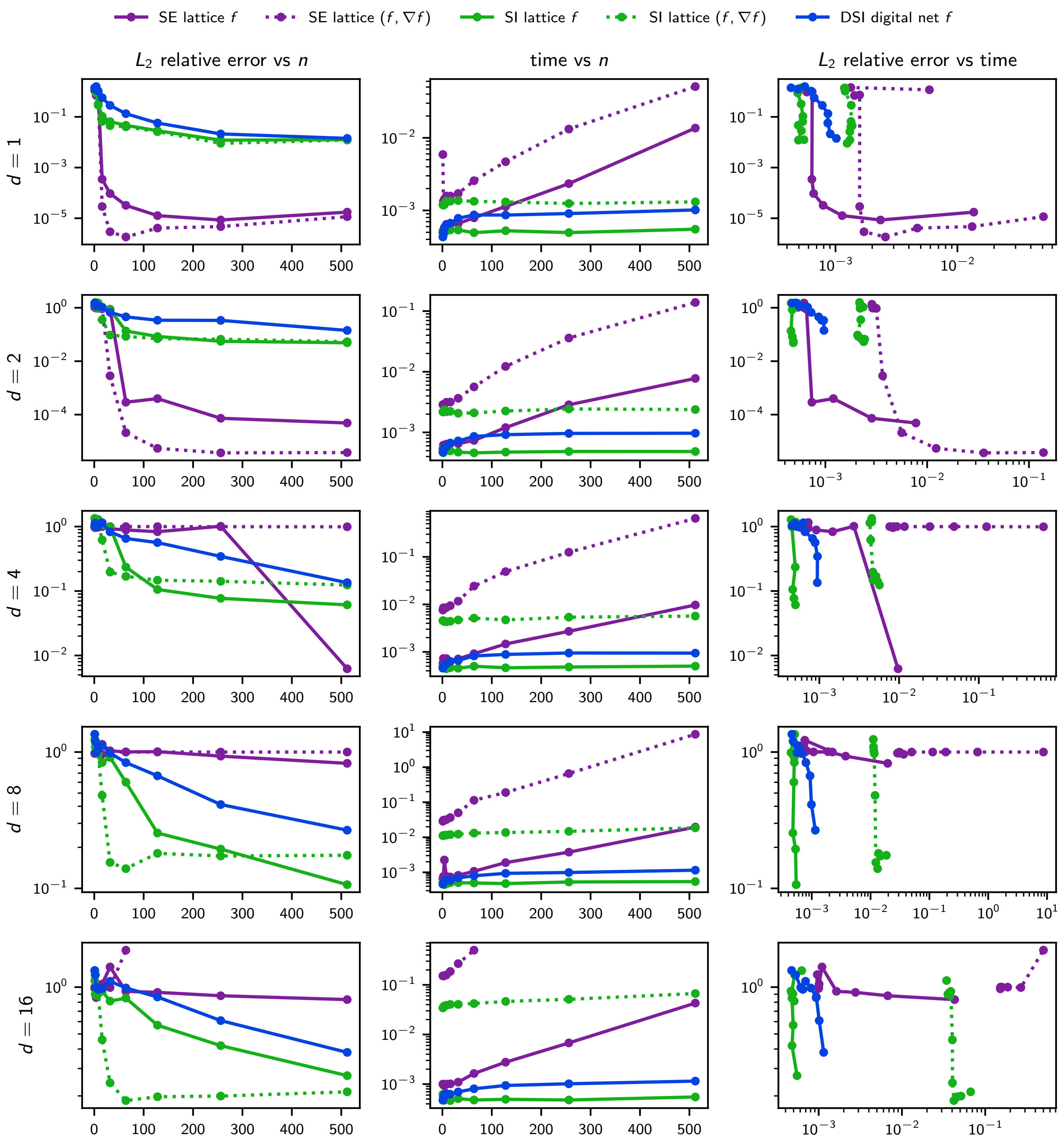

Figure 9: GPR of varying dimension d on the Styblinski–Tang benchmark function. Here we compare the error, number of samples n, time per optimization step, and inclusion of gradient information ∇f. GPR was run for 250 optimization steps.