# Deep Gaussian Process Surrogates for Computer Experiments

Annie E. Sauer

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Statistics

Robert B. Gramacy, Chair

David Higdon

Marco A. R. Ferreira

Jennifer Van Mullekom

April 12, 2023

Blacksburg, Virginia

Keywords: emulator, non-stationary, uncertainty quantification,

active learning, Vecchia approximation

# Deep Gaussian Process Surrogates for Computer Experiments

Annie E. Sauer

(ABSTRACT)

Deep Gaussian processes (DGPs) upgrade ordinary GPs through functional composition, in which intermediate GP layers warp the original inputs, providing flexibility to model non-stationary dynamics. Recent applications in machine learning favor approximate, optimization-based inference for fast predictions, but applications to computer surrogate modeling – with an eye towards downstream tasks like Bayesian optimization and reliability analysis – demand broader uncertainty quantification (UQ). I prioritize UQ through full posterior integration in a Bayesian scheme, hinging on elliptical slice sampling of latent layers. I demonstrate how my DGP's non-stationary flexibility, combined with appropriate UQ, allows for active learning: a virtuous cycle of data acquisition and model updating that departs from traditional space-filling designs and yields more accurate surrogates for fixed simulation effort. I propose new sequential design schemes that rely on optimization of acquisition criteria through evaluation of strategically allocated candidates instead of numerical optimizations, with a motivating application to contour location in an aeronautics simulation. Alternatively, when simulation runs are cheap and readily available, large datasets present a challenge for full DGP posterior integration due to cubic scaling bottlenecks. For this case I introduce the Vecchia approximation, popular for ordinary GPs in spatial data settings. I show that Vecchia-induced sparsity of Cholesky factors allows for linear computational scaling without compromising DGP accuracy or UQ. I vet both active learning and Vecchia-approximated DGPs on numerous illustrative examples and real computer experiments. I provide open-source implementations in the `deepgp` package for `R` on CRAN.

# Deep Gaussian Process Surrogates for Computer Experiments

Annie E. Sauer

(GENERAL AUDIENCE ABSTRACT)

Scientific research hinges on experimentation, yet direct experimentation is often impossible or infeasible (practically, financially, or ethically). For example, engineers designing satellites are interested in how the shape of the satellite affects its movement in space. They cannot create whole suites of differently shaped satellites, send them into orbit, and observe how they move. Instead they rely on carefully developed computer simulations. The complexity of such computer simulations necessitates a statistical model, termed a "surrogate", that is able to generate predictions in place of actual evaluations of the simulator (which may take days or weeks to run). Gaussian processes (GPs) are a common statistical modeling choice because they provide nonlinear predictions with thorough estimates of uncertainty, but they are limited in their flexibility. Deep Gaussian processes (DGPs) offer a more flexible alternative while still reaping the benefits of traditional GPs. I provide an implementation of DGP surrogates that prioritizes prediction accuracy and estimates of uncertainty. For computer simulations that are very costly to run, I provide a method of sequentially selecting input configurations to maximize learning from a fixed budget of simulator evaluations. I propose novel methods for selecting input configurations when the goal is to optimize the response or identify regions that correspond to system "failures". When abundant simulation evaluations are available, I provide an approximation which allows for faster DGP model fitting without compromising predictive power. I thoroughly vet my methods on both synthetic "toy" datasets and real aeronautic computer experiments.

# Acknowledgments

I was blessed to be surrounded with amazing mentors, friends, and family during my time at Virginia Tech. I am grateful for all those who played a part in my graduate school journey, but especially the following people.

First and foremost, my advisor Bobby. He not only taught me how to "research" (whatever that means), but also how to teach, mentor, code, write technically (and colloquially), perservere through the roadblocks, and give myself grace when I inevitably make mistakes. His financial support – and encouragement to put myself out there – allowed me to travel the world and present my work. He is truly a baller in this field, and I hope to get on his level one day.

Dave Higdon, for believing a young second year student like me could tackle these deep Gaussian processes in the first place, and for always bringing a positive energy and bright outlook to our discussions.

Jim Warner, who gave me the opportunity to work for NASA not once, but twice (on short notice too). Although I didn't know it at the time, my work with NASA would end up inspiring a lot of this dissertation.

Jean D. Gibbons, for her generous financial support. Her fellowship made the choice to come to graduate school at Virginia Tech a no-brainer.

My "fishbowl" squad - Jake, Quyen, and Ryan. My grad school journey was shaped by those first years, studying for the qualifying exams and figuring out what we wanted to do with our lives.

Last but not least, my fiancé Jon and my amazing family, for cheering me on and having my back every step of the way.

# Contents

# Chapter 1

# Introduction

Virtualization and simulation increasingly play a fundamental role in the design and study of complex systems that are either impossible or infeasible to experiment with directly. Examples abound in epidemiology [e.g., 31], engineering [e.g., 133], aeronautics [e.g., 86], economics [e.g., 23, 71], and ecology [e.g., 58], to name a few. Simulation can provide insight into complex processes that might otherwise be immeasurable, but it may come with significant computational costs. Such cases necessitate statistical surrogates, meta-models that furnish accurate predictions for nonlinear dynamics with appropriate uncertainty quantification (UQ) from a limited, carefully designed simulation campaign. Surrogates may thus stand-in for actual simulations at untried input settings to support downstream tasks such as calibration [56, 70], optimization [60, 98], and sensitivity analysis [83, 94, 107]. Surrogate accuracy, combined with effective UQ, is key to the success of such enterprises.

Gaussian processes (GPs) are a common surrogate modeling choice [42, 104, 109] because they offer accurate nonlinear predictions and UQ in a semi-analytic Bayesian nonparametric framework. Despite their prowess in many settings, the typical assumption of stationarity – made primarily for computational convenience – compromises the GP's ability to accommodate features common to many computer simulations, such as regime changes in input-output dynamics. Early solutions to this problem from a spatial statistics [e.g., 55, 96, 108, 113] and machine learning (ML) perspective [e.g., 102, 103] focused on low input dimension (e.g., longitude and latitude) and small training data sizes. More recent advances in non-stationary

spatial modeling [e.g., 12, 64] emphasize scaling-up to larger training data sets, still privileging low-dimensional input spaces. The computer surrogate modeling literature now has bespoke solutions which work well in specific, modest-dimensional cases: treed-GP hybrids for when non-stationarity manifests along coordinate directions [44]; composite processes when it arises as changes in amplitudes [2]; local data subsets when training data are massive [16, 43]. Yet general purpose design and modeling strategies remain elusive.

I see promise in the form of deep Gaussian processes (DGPs) – originating in geo-spatial communities [108, 113] but recently popularized by Damianou and Lawrence [20] with analogy to deep neural networks. The setup neatly synthesizes elements of warping, latent inputs, and composition as earlier introduced in isolation by authors of several of the papers cited above. By warping the input space through hidden Gaussian layers, effectively moving some training samples closer and others farther apart, they achieve non-stationarity even under otherwise conventional kernel structures. Prowess has been demonstrated on many classification tasks [20, 32, 131]. Application as surrogates for simulation experiments is rather less well developed [100].

The compositional form of the DGP likelihood makes direct Bayesian inference impossible. Initial schemes leveraged approximate variational inference [e.g., 13, 20, 106], embracing computational thrift at the cost of full UQ. This is a mismatch to typical surrogate modeling scenarios in which downstream applications necessitate careful, thorough uncertainty quantification. I thus depart from the canonical inferential apparatus to favor Markov chain Monte Carlo (MCMC) posterior sampling via a novel hybrid Gibbs-Metropolis and elliptical slice sampling [ESS; 91] scheme. This is computationally heavy, but adds value in straightforward implementation, and remains tractable in typical surrogate modeling scenarios.

In this dissertation I provide a variety of examples demonstrating DGP prowess on accuracy and UQ. When regimes change abruptly, DGPs mimic partition schemes. When dynamics

change more smoothly, they mimic kernel evolution. When stationary, they gracefully revert to ordinary GP dynamics via identity warpings. But there is a catch. Non-stationary flexibility is intimately twinned with training data size and structure; you need enough of the right kind of data to detect relevant dynamics. One solution is to specifically target areas of interest through active learning (AL): the sequential design and build-up of a surrogate through a virtuous cycle of data acquisition and fitting. AL is crucial when the simulator is computationally expensive and training data is limited. [In 2021, I collaborated with engineers at NASA Langley Research Center on a proprietary simulator that required nearly a week of compute time to produce a single run, even with careful utilization of supercomputing resources.]

In lieu of a targeted research objective, sequential designs simply seek to maximize learning with each successive evaluation of the simulator. This was the goal in one of my motivating real-world experiments: the *Test Particle Monte Carlo* simulator [86] developed by researchers at Los Alamos National Lab to simulate satellites moving through low earth orbit (featured later in Sections 3.1.5 and 4.5). Researchers aimed to achieve under 1% root mean squared prediction error with as few evaluations of the simulator as possible. Such goals boil down to AL acquisition criteria that minimize posterior predictive variance. Some inroads have been made with DGPs in these realms. Dutordoir et al. [25] fit DGPs to sequentially collected data, but acquisition criteria were not based on the DGP fits. Rajaram et al. [101] suggested a maximum variance criterion, a strategy sometimes called "active learning MacKay" [ALM; 80], with DGPs. ALM is under-powered compared to aggregate variance-based criteria that are more common in the computer experiments literature. I propose using integrated mean-squared error (IMSE) and its ML cousin "active learning Cohn" [ALC; 15], which has better properties and empirical performance under GPs [45, 115].

Alternatively, the objective may be to find the input configuration that optimizes the re-

sponse variable, so-called "global optimization". As an example, consider the "QBLADE" simulator [84] which simulates the energy output of a wind turbine based on blade design and environmental circumstances. Naturally, finding the design that yields the highest energy output is of great interest. The expected improvement criterion [EI; 60] selects inputs by balancing exploitation (inputs that are expected to yield high responses) and exploration (inputs with high predictive variance). Sequential design using EI is commonly termed "Bayesian optimization" (BO). Hebbal et al. [53] applied DGPs to BO but relied on approximate variational inference for DGP fitting. The challenge of DGP-based BO in my MCMC setting lies in the optimization of the EI criteria, which is typically multi-modal with many local maxima. Multi-start numerical optimizers are not feasible for thousands of DGP posterior samples. Pseudo-optimization through evaluation over a set of candidate inputs is required. For this setting, I propose a new candidate allocation scheme hinging on Delauney triangulation of the training design and show that it outperforms traditional space-filling candidates.

Another common surrogate modeling objective, which is especially prevalent in aerospace engineering applications, is to identify a specific contour or level set in the response surface. For example, consider a simulation of an aircraft gliding through the atmosphere at varying speeds and configurations [such as the Common Research Model developed by NASA, 127]. The simulator outputs the amount of vibration experienced by the aircraft. The research objective is to identify which inputs result in unsafe conditions, i.e., vibration that exceeds safety thresholds. Ensuring inputs pass safety standards with acceptably low failure probabilities is termed "reliability analysis" [see 119, for work I did with engineers at NASA on reliability analysis with gradient-enhanced GPs]. Targeting an entire level set is even trickier than targeting a single optimum, and as far as I know there is no existing work utilizing DGPs for contour location. Entropy is a popular acquisition criteria for typical GP surrogates in

a contour finding setting [82], but it is extremely peaky and underweights exploration of areas with high uncertainty. Cole et al. [17] circumvented these limitations with a clever hybrid-local numerical optimization, but again such schemes are not feasible for MCMC-based DGP surrogates. I propose a new candidate-based scheme that extends my Delauney triangulation candidates to select acquisitions on the "Pareto front" of entropy and prediction uncertainty. The candidate-based nature of my proposed contour location algorithm allows for the application of DGPs to previously untouched realms. This work is motivated by application to an SU2 computational fluid dynamics simulation of incrompessible flow around an airfoil [28].

In active learning applications like the three discussed above, computational bottlenecks center on the number of evaluations of the computer simulation. If instead a large training data set is readily available, computational concerns shift to those of fitting and evaluating the surrogate. Inference for GPs requires the evaluation of multivariate normal likelihoods, which involves dense matrix decompositions that scale cubically with the size of the training data. Computer simulation campaigns used to be small, but recent advances in hardware, numerical libraries, and STEM training have democratized simulation and led to massively larger campaigns [e.g., 69, 75, 79, 81, 123]. The literature has since adapted to address this bottleneck by borrowing ideas from machine learning and geo-spatial GP approximation. Examples include sparse kernels [87], local approximations [16, 30, 43], inducing points [35, 99], and random feature expansions [81]. See [52] and [77] for thorough reviews. Here I am drawn to a family of methods that leverage "Vecchia" approximation [128], which imposes a structure that generates a sparse Cholesky factorization of the precision matrix [21, 66]. When appropriately scaled or generalized [22, 65, 121, 122], and matched with sparse-matrix and multi-core computing facilities, Vecchia-GPs dominate competitors [67] on the frontier of accuracy, UQ, and speed.

Computational costs are even more significant in DGP settings. Cubic scaling in flops is present in multitude, with large dense matrices at each latent DGP layer. MCMC sampling exacerbates cubic bottlenecks and limits training data sizes to the hundreds. Recent works on DGPs have overwhelmingly embraced inducing point approximations [117] to circumvent computational obstacles [e.g., 13, 20, 51, 106], but these inducing point approximations are notoriously low fidelity. Instead, I embrace Vecchia approximation which offers higher fidelity approximations without the added burden of selecting inducing point locations [129]. Incorporating a Vecchia approximation at each DGP layer allows for computation that scales linearly with training data size, thereby expanding the utility and reach of fully-Bayesian posterior inference for DGP surrogate models.

In the remainder of this chapter, I offer a birds-eye view of my contribution and provide a necessary review of GP fundamentals. In Chapter 2, I detail my fully-Bayesian DGP model. In Chapter 3, I develop active learning methodology for variance reduction, Bayesian optimization, and contour location. That chapter culminates in a motivating application to an SU2 computational fluid dynamics airfoil computer experiment. In Chapter 4, I integrate Vecchia approximation into my Bayesian DGP framework to accommodate larger data sizes. Each chapter (or section in the case of Chapter 3) begins with review of existing methodologies, progresses into details of my proposed advancements, and wraps up with implementation details and empirical results.

## 1.1 Summary of Contribution

My goal is to develop and promote deep Gaussian processes as surrogate models for computer experiments (if it wasn't obvious enough from my title). This requires developing new methodology, perfecting straight-forward and feasible implementations, providing pub-

licly available and easy-to-use software, and demonstrating superior performance in both simulated and real-world scenarios. Although I am primarily motivated by applications to aerospace engineering, I believe this work will have a lasting long-term impact on a vast array of scientific fields. The use of computer simulations is continually expanding, and I suspect this trend will only escalate over time.

Gaussian processes have a tight grip on the surrogate modeling community, and many have accepted that their stationary limitations are worth the nonlinear predictive prowess and analytic UQ that comes with them. But regime shifts are prevalent in computer experiments, and stationarity presents a clear handicap in these situations. Previous, more flexible GP alternatives such as deep neural networks [125] and Bayesian additive regression trees [14] have not been widely embraced, most likely a consequence of their less-than-thorough accounts of uncertainty. There is a great need for a flexible surrogate model that can outperform GP accuracy and UQ in the face of non-stationary dynamics.

First, I propose a new inferential apparatus for deep Gaussian process regression models that prioritizes uncertainty quantification through full posterior integration of latent variables in a Bayesian MCMC sampling scheme. I sample latent Gaussian layers through elliptical slice sampling [ESS; 91]. Previously utilized Metropolis-Hastings sampling of these layers suffers from sticky chains/poor mixing and is not feasible for input dimensions above one or two [113]. I suspect the lack of an efficient sampling scheme is the reason DGPs had not yet made their way from the ML community (where approximate variational inference was sufficient) to the surrogate modeling community. On the contrary, the Gaussian proposals and rejection-free nature of ESS is uniquely suited for the multi-modal posteriors of DGP latent layers. Perhaps most importantly, I provide an open-source off-the-shelf implementation of my fully-Bayesian DGP in the `deepgp` R-package on CRAN [110].

Second, I incorporate variance-reducing acquisition criteria for sequential designs with DGPs

when training data is limited (which is the case when the simulator is computationally expensive). I detail implementations of both integrated mean squared error and its counterpart "active learning Cohn" [15], which account for both variance in posterior predictions and uncertainty in the warpings (i.e. the latent layers) of the DGP. When paired with a typical stationary GP, these criteria result in space-filling designs, nullifying the efforts of a strategic sequential design. But when combined with the non-stationary flexibility of a DGP and full posterior integration, they allow for adaptive designs that depart from traditional space-filling schemes and outperform both stationary and non-stationary competitors. These criteria are also integrated into the `deepgp` package.

Third, I employ the expected improvement criteria to allow for DGP-based Bayesian optimization. The key contribution here is the integration of the DGP with strategic selection of candidate locations through Delauney triangulation, an idea originating from joint work with my advisor Bobby Gramacy and fellow student Nathan Wycoff [46]. Although space-filling candidates sufficed for optimization of the variance-reducing criteria mentioned above, there was much room for improvement in the placement of candidates. Strategically allocating candidates between existing training data allows for superior acquisitions, yielding better performance for fixed simulation effort – all while avoiding any bulky numerical optimization. Integration of the EI criterion with my Bayesian DGP is provided as an optional feature in the `deepgp` package.

Fourth, I propose an avenue for utilizing Bayesian DGP surrogates in reliability analysis with real-world implications to aerospace systems design. To my knowledge, this is the first work to apply DGPs in a contour location setting. This work is perhaps the most exciting development as it is the culmination of several of my earlier contributions: fully-Bayesian DGPs, sequential design, and Delauney triangulation candidates. I develop a novel acquisition criteria that balances entropy and posterior variance to strike a balance between

exploitation and exploration. I provide evidence of excellent performance on a real-world aerospace simulation.

Finally, I incorporate the Vecchia approximation into my DGP apparatus to allow for applications to upwards of 100,000 simulator evaluations. My original DGP implementation was limited to data sizes in the hundreds, so the expansion to a hundred thousand is a massive extension. I present empirical comparisons of my Vecchia-DGP to both stationary Vecchia GP's and previously considered "state-of-the-art" non-stationary DGPs, which utilize a variety of inferential tools including variational inference and Hamiltonian Monte Carlo sampling [9]. I demonstrate Vecchia-DGP superiority on a variety of non-stationary examples.

All of my contributions are supported by the `deepgp` package and a public git repository of reproducible examples:

https://bitbucket.org/gramacylab/deepgp-ex/.

I will highlight some package functionality throughout, but thorough demonstration and documentation is provided in the package vignette and reference manual on CRAN [110]. At the time of writing, much of this work (but not all) has been published in peer-reviewed journals [46, 111, 112].

## 1.2 Foundations: Gaussian Process Surrogates

Let $f : \mathbb{R}^d \to \mathbb{R}$ represent a (possibly noisy) black-box function, say to abstract a computer model simulation. To help manage simulation costs, it is common to develop a surrogate $\hat{f}_n : \mathbb{R}^d \to \mathbb{R}$ that approximates $f$ based on outputs from a small set of $n$ designed inputs. Let $X$ denote an $n \times d$ training design of input locations and $Y = f(X)$ denote the corresponding function evaluations of size $n \times 1$. [In Chapter 3, I will denote $X = X_n$ and $Y = Y_n$ to

emphasize the training data size when building up sequential designs.] Throughout I use lowercase $x_i$ to refer to the $i^{\text{th}}$ (transposed) row of $X$, and likewise for $Y$.

The canonical GP surrogate assumes a multivariate normal distribution (MVN) over the response,

$$Y \sim \mathcal{N}_n\left(\mu, \Sigma(X)\right). \tag{1.1}$$

To streamline notation, denote $\Sigma_n = \Sigma(X)$. The mean $\mu$ may be linear in columns of $X$, but $\mu = 0$ is often sufficient after centering [42, 109].

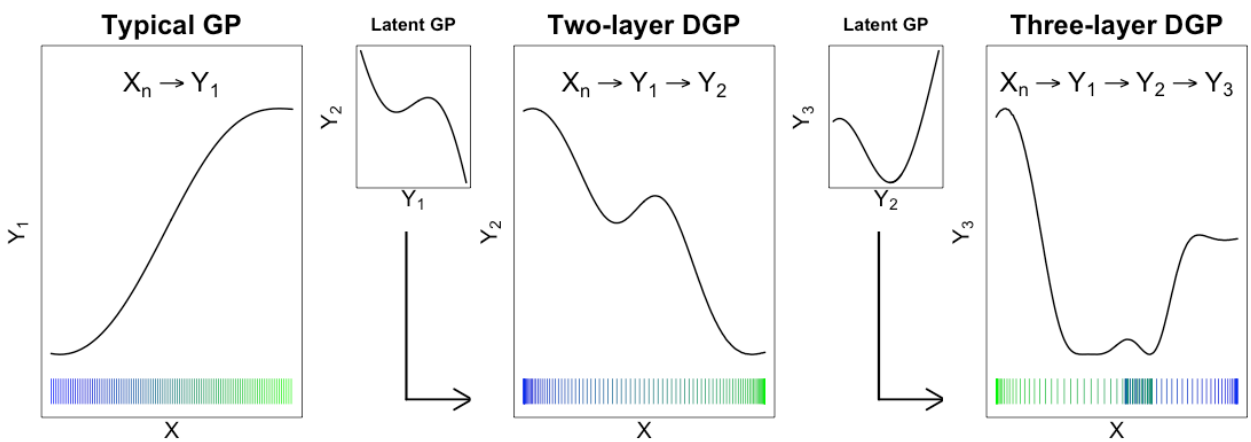


Figure 1.1: GP prior realization passed through latent GPs to generate two- and three-layer DGPs. Colored tick marks on the x-axes show the distribution of the latest input ($X$, $Y_1$, and $Y_2$). $Y_1$ clusters data at the boundaries, and $Y_2$ overlaps data near the middle of the input space.

The left panel of Figure 1.1 shows a "prior" random draw from such an MVN based on a gridded $X$ with $d = 1$ under a covariance structure determined by inverse (squared) Euclidean distance:

$$\Sigma_n^{ij} = \tau^2 \left( \exp\left( -\frac{||x_i - x_j||^2}{\theta} \right) + g\mathbb{I}_{\{i=j\}} \right). \tag{1.2}$$

In this so-called isotropic Gaussian kernel, $\tau^2$, $\theta$, and $g$ act as hyperparameters controlling the scale, correlation strength (lengthscale), and noise level (nugget) respectively. More generally, I will often denote $\Sigma_n = \tau^2 \left(K_\theta(X) + g\mathbb{I}_n\right)$ where $K_\theta^{ij} = k\left(||x_i - x_j||^2/\theta\right)$ for kernel

$k(\cdot)$. Any choice of $k(\cdot)$ leading to positive definite $\Sigma(X)$ is valid. Popular choices include the squared exponential as in Eq. (1.2) and Matèrn [120], but my work here is not kernel specific. [I offer both kernels in my software.]

There are many variations on this theme. For example, vectorized $\theta$ allow the rate of decay of correlation to vary with input direction. Such embellishments implement a form of *anisotropy*, a term preferred in geo-spatial contexts, or *automatic relevance determination* in machine learning [78]. My DGP setup does not need such modifications; once latent layers are involved, such flexibility manifests more parsimoniously via those values. Several of my competitors in later sections do utilize additional hyperparameters and/or equivalent affine pre-scaling [130].

Given (potentially noisy) training data samples $D_n = \{X, Y\}$, I would want to learn these hyperparameters. The MVN model structure emits the following (marginal) log likelihood,

$$\log \mathcal{L}\left(Y \mid X\right) \propto -\frac{1}{2}\log|\Sigma_n| - \frac{1}{2}Y^\top \Sigma_n^{-1} Y. \tag{1.3}$$

Maximum likelihood estimates are commonly used as plug-ins for $\tau^2$, $\theta$, and $g$. MLE $\hat{\tau}^2$ has a closed form. Derivative-based numerical maximization is required for $\hat{\theta}$ and $\hat{g}$. Details and implementation are provided by Gramacy [e.g., 42, Chapter 5]. Here I promote Bayesian posterior sampling, thinking ahead to DGPs. One can marginalize $\tau^2$ out of the posterior analytically under a reference prior ($\pi(\tau^2) \propto 1/\tau^2$), or any conditionally conjugate inverse Gamma specification; however, $\theta$ and $g$ require rejection-based MCMC schemes (i.e., Metropolis-Hastings). Details are provided, e.g., by Gramacy [42, Section 5.5]. When training data sizes are low, it can be crucial to average over posterior uncertainty for $\theta$ and $g$ which, together, facilitate a signal-to-noise trade-off. For deterministic (or very low noise) computer model simulations, I usually fix $g$ at a small constant, e.g., $g = 10^{-6}$. Regardless

of these choices, evaluation of the likelihood (1.3) relies on both the inverse and determinant of $\Sigma_n$. For a dense $n \times n$ matrix, this is an $\mathcal{O}(n^3)$ operation with conventional libraries.

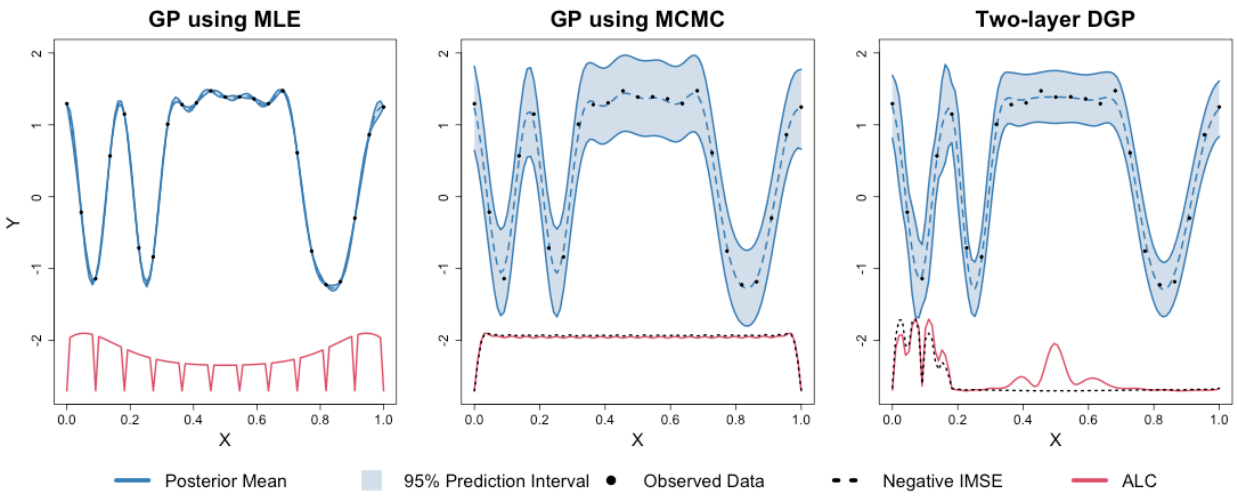


Figure 1.2: Surrogates $\hat{f}_n$ fit to data from $f(x)$ in Eq. (3.4): (left) using MLE $\hat{g}$ and $\hat{\theta}$; (center) using MCMC sampling of $g$ and $\theta$; (right) using a two-layer DGP (described in Section 2.1). ALC and negative IMSE (described in Section 3.1) are plotted along a dense grid of candidate locations.

Given $D_n$, and under settings of hyperparameters (either MLE or via posterior sampling), the posterior predictive distribution for an $n' \times d$ matrix of testing locations $\mathcal{X}$ has closed form

$$Y(\mathcal{X}) \mid D_n \sim \mathcal{N}_{n'}\left(\mu_Y(\mathcal{X}), \Sigma_Y(\mathcal{X})\right), \quad \text{where} \quad \begin{aligned} \mu_Y(\mathcal{X}) &= \Sigma(\mathcal{X}, X)\Sigma_n^{-1}Y \\ \Sigma_Y(\mathcal{X}) &= \Sigma(\mathcal{X}) - \Sigma(\mathcal{X}, X)\Sigma_n^{-1}\Sigma(X, \mathcal{X}), \end{aligned} \tag{1.4}$$

and $\Sigma(X, \mathcal{X})$ is an $n \times n'$ matrix derived by extending the kernel across training and testing locations. These equations, coupled with an inferential strategy (e.g., MLE $\hat{\tau}^2$ converts Gaussian to Student-$t$), complete the description of surrogate $\hat{f}_n$. A 1d illustration is provided in the left and center panels of Figure 1.2, based on MLE and full posterior sampling, respectively. The GP surrogate's flexibility provides accurate posterior mean estimates in the non-linear regions, but MLE point-estimates (left) lead to over-fitting – interpreting noise

as signal – in the flat region. Full posterior sampling (center) appropriately averages over posterior evidence for both regimes.

The above GP specification is *stationary* because only relative positions of training and testing inputs are involved (1.2). Consequently, identical input-output dynamics apply everywhere in the input space, which can be limiting for some computer simulations. A great example comes from aeronautics/computational fluid dynamics. Lift forces on aircraft are fundamentally different at high speed versus low, and in particular at the sound barrier where the transition is abrupt [97]. This so-called stationarity limitation is coupled, from a practicality perspective, with computational bottlenecks. Even if stationarity is relaxed, multiple regimes may not be recognized without large amounts of training data.

# Chapter 2

# Deep Gaussian Process Surrogates

A deep Gaussian Process [20] is a hierarchical layering of GPs where each layer is conditionally multivariate normal (MVN). Dunlop et al. [24] detailed four methods for constructing DGPs, each providing different levels of feasibility and interpretability. I view DGPs as functional compositions, arguably the most interpretable and easily implemented option.

In a DGP prior, the inputs $X$ are mapped through one or more intermediate GPs before reaching the response $Y$. The inputs to those intermediate "layers" act as latent variables, warping the input space into a plausibly stationary regime but remaining unobserved. Henceforth, I shall refer to typical GP regression as described in Section 1.2 as a "one-layer" GP. A two-layer model, with inputs to the new hidden GP labeled as $W$, may be described hierarchically as

$$Y \mid W \sim \mathcal{N}_n\left(0, \Sigma(W)\right), \qquad W_k \overset{\text{ind}}{\sim} \mathcal{N}_n\left(0, \Sigma_k(X)\right), \quad k = 1, \ldots, p. \tag{2.1}$$

$W = [W_1, \ldots, W_p]$ is an $n \times p$ matrix with each row representing one observation and each column representing a dimension of the latent space. I adopt the deep learning convention of referring to the component dimensions of $W$ as "nodes". Each node has its own $\Sigma_k(X)$ which may or may not share features, like kernels and hyperparameters, with others.

Adding additional depth and nodes to intermediate layers will eventually yield diminishing returns. Damianou and Lawrence [20] found some justification for a five-layer DGP in clas-

sification tasks, but two- and three-layer DGPs have been sufficient for real-valued outputs common to computer surrogate modeling [100]. Dunlop et al. [24] similarly preferred two and three layers. I find that low numbers of latent nodes (no higher than the dimension of the input space) and limited depth (no deeper than three layers) are sufficient for surrogate modeling. More details are provided in Section 2.1, with evidence in Chapter 3. Again, my work and implementation are not limited to this choice.

Conditional on $\Sigma(\cdot)$ and each $\Sigma_k(\cdot)$, the marginal likelihood may be represented as an integral over unknown $W$,

$$\mathcal{L}\left(Y \mid X\right) = \int \mathcal{L}\left(Y \mid W\right) \prod_{k=1}^{p} \mathcal{L}\left(W_k \mid X\right) \, dW, \tag{2.2}$$

where $\log \mathcal{L}(Y \mid W)$ and $\log \mathcal{L}(W_k \mid X)$ are defined as in Eq. (1.3). A three-layer model will have two hidden layers (denoted $Z$ and $W$) and will involve three MVN likelihoods with a double integral over $Z$ and $W$. The center and right panels of Figure 1.1 show the hierarchical composition of two- and three-layer DGPs in a single dimension, accumulating from the left panel's ordinary, one-layer GP. As uniformly distributed $X$ are fed through intermediate layers they are re-distributed, and response values are no longer stationary.

The benefit of DGPs from the perspective of active learning (AL) is foreshadowed in the right panel of Figure 1.2. A degree of non-stationarity, as facilitated by warping the input space so that some pairs are (effectively) closer than others when calculating covariance, is sufficient to nudge acquisitions towards the "interesting" part of the input space, rather than being essentially space-filling.

Closed form Bayesian inference for two- and three-layer DGPs is intractable since it is not possible to analytically integrate the hidden layer(s) out of the (marginal) likelihood (2.2). Variational inference and other maximization-oriented methods [20, 106] may be computationally thrifty, but they yield probabilistic representations which can over-simplify, partic-

ularly as regards UQ. MCMC methods address uncertainty more thoroughly at the expense of computation. To mitigate these costs, efficient mixing of the Markov chain is essential.

## 2.1 Model Specification

In a DGP, extreme flexibility can come at the cost of identifiability and practicality. Here I provide a template that has worked well for surrogate modeling in several realistic settings, as showcased in Sections 3.1.4, 3.1.5, 3.3.4, and 4.5, and which supports downstream tasks such as AL. I begin with a two-layer setup (2.1), defining the hierarchical structure in terms of distance-based covariance (1.2).

$$\begin{aligned} Y \mid W &\sim \mathcal{N}_n\left(0, \tau^2(K_{\theta_y}(W) + g\mathbb{I}_n)\right) \\ W_k &\overset{\text{ind}}{\sim} \mathcal{N}_n\left(0, K_{\theta_w[k]}(X)\right), \quad k = 1, \ldots, p \end{aligned} \tag{2.3}$$

Notice that scale $\tau^2$ and nugget $g$ are placed on the outer layer only. Specifying noiseless hidden layers with unit scale is crucial, I find, to stable posterior inference. I find that $p = \dim(W) = \dim(\mathrm{X}) = d$ works well as a default, although smaller values may be appropriate for high-dimensional $X$. Note that, like a column of $X$ or response $Y$, $W_k$ is an $n$-vector with one coordinate for each training data point. Although nodes of $W$ could be modeled jointly with cross covariances [113], I recommend the following simplifications.

i. $W_j$ and $W_k$, for $j \neq k$, are conditionally independent [20].

ii. Each $W_k$ is modeled as isotropic (1.2) in inputs $X$ via unique scalar lengthscale $\theta_w[k]$, for $k = 1, \ldots, p$, regardless of the input dimension $d$.

iii. Similarly, $Y$ is modeled as isotropic in all nodes of $W$ with scalar lengthscale $\theta_y$.

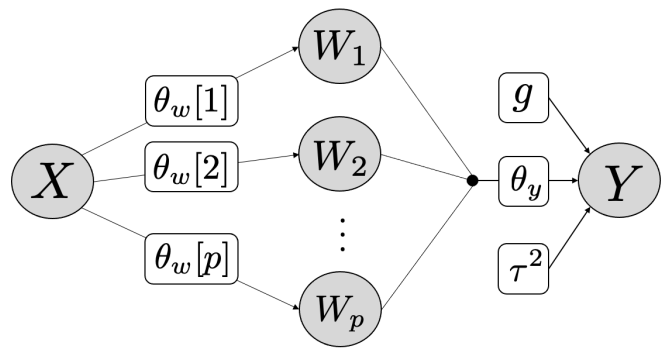


Figure 2.1: Model structure for a two-layer DGP with $p$ latent nodes.

Figure 2.1 depicts this model structure, still focusing on the two-layer setup. Impositions ii.–iii. are essential to reign in complexity. Latent $W$, even with a single node but especially for $p \gg 1$, offer more than enough flexibility to "imitate" a separable/anisotropic kernel without the added complexity of additional lengthscale parameters. This is simplest to see with $p = d$. If the data desire lower correlation in one coordinate direction compared to another, then components of $W$ can organize themselves so that $W_j$, say, "fans out" more than $W_k$ does. Of course, it is easy to imagine more complicated arrangements. The filled circle between the $W$ and $Y$ layers in the diagram represents imposition iii., depicting that all nodes of $W$ form inputs to the outer layer for $Y$ under isotropic (scalar) lengthscale $\theta_y$.

Building off that setup, consider a three-layer model built as follows.

$$
\begin{aligned}
Y \mid W &\sim \mathcal{N}_n\left(0, \tau^2(K_{\theta_y}(W) + g\mathbb{I}_n)\right) \\
W_k \mid Z &\overset{\text{ind}}{\sim} \mathcal{N}_n\left(0, K_{\theta_w[k]}(Z)\right), \quad k = 1, \ldots, p \\
Z_j &\overset{\text{ind}}{\sim} \mathcal{N}_n\left(0, K_{\theta_z[j]}(X)\right), \quad j = 1, \ldots, p
\end{aligned}
$$

Here, as above, conditional independence and isotropy (i.-ii.) for $W$ is duplicated in the new latent layer $Z$. I have found it helpful to restrict the number of nodes in $Z$ to match $W$, i.e., $\dim(Z) = \dim(W) = p \leq d$. Again, each $Z_j$ is an $n$-vector of latent variables. Figure 2.2 depicts this structure diagrammatically. The filled circle between $Z$ and $W$ layers, with

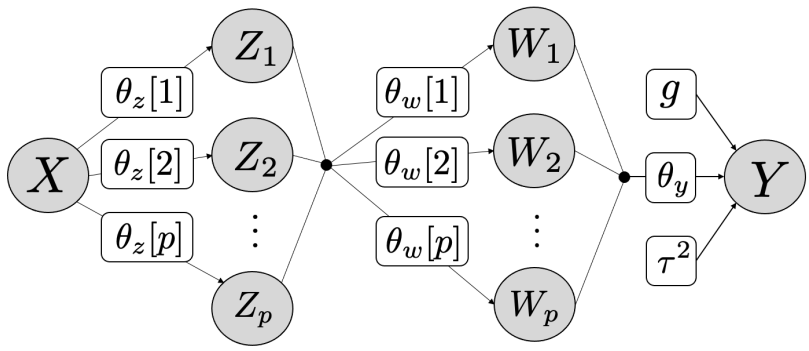


Figure 2.2: Model structure for a three-layer DGP with $p$ latent nodes in $Z$ and $W$.

edges fanning out in both directions, is intended to represent communication between all $Z$ and $W$ nodes, like an "information bus" capturing $p^2$ interconnections. Each latent $Z_j$ input, $j = 1, \ldots, p$, is connected to each $W_k$ output through an inverse-distance-based covariance kernel with (scalar) lengthscale parameters $\theta_w[k]$, for $k = 1, \ldots, p$.

### Hyperparameter priors

I complete my Bayesian hierarchical model specification with the following priors on hyperparameters, comprising lengthscales $\theta = \{\theta_y, \theta_w, \theta_z\}$, scale $\tau^2$, and nugget $g$ on the outer layer. Conditionally conjugate inverse Gamma (IG) priors are common for scales like $\tau^2$ because they may subsequently be analytically integrated out of an otherwise numerical posterior sampling scheme. Details are provided in Section 2.2 shortly. Specifying IG parameters of zero [following the description in 38] leads to a so-called reference prior $\pi(\tau^2) \propto 1/\tau^2$ [8], which is a common default in GP spatial modeling. Although improper, the posterior is proper (in my zero-mean context) as long as $n \geq 1$, i.e., as long as there is at least one training data example.

I prefer proper priors for the other parameters and choose members of the Gamma family as their familiar form offers convenient specification, which I tailor to penalize against pathological settings such as $\theta \to \{0, \infty\}$. Specifically, consider $\{\theta, g\} \overset{\text{iid}}{\sim} \text{G}(3/2, b_{[\cdot]})$ where $b_{[\cdot]}$

is adjusted depending on the hyperparameter in question. This choice mirrors other work in Bayesian posterior sampling of GP hyperparameters [e.g., 44] and in penalized marginal maximum likelihood settings [e.g., 43]. I find it helpful to nudge the posterior toward a hierarchy in the latent nodes with $b_{[\theta_y]} > b_{[\theta_w]} > b_{[\theta_z]}$ encoding a prior belief that as layers get deeper they should be less "wiggly" (closer to the identity mapping). Particular, user-adjustable, default values for $b_{[\cdot]}$ provided in my software are detailed in Section 2.4. When modeling a deterministic computer model simulation, I fix $g = \varepsilon$, a small non-zero value such as `sqrt(.Machine$double.eps)` in R, to interpolate training data.

## 2.2 Posterior Sampling

Here I propose a hybrid Gibbs-ESS-Metropolis scheme for sampling hyperparameters and latent layers. A closed form GP marginal likelihood, integrating out scale $\tau^2$ under the reference prior, is crucial to the subsequent development. Although the calculation is rather textbook, e.g., see Gramacy [42, Section 5.5, exercise 1], I provide it below in my notation for concreteness.

$$\log \mathcal{L}(Y \mid W, \theta_y, g) \propto -\frac{n}{2} \log\left(n\hat{\tau}^2\right) - \frac{1}{2} \log |K_{\theta_y}(W) + g\mathbb{I}_n| \quad \text{with} \quad \hat{\tau}^2 = \frac{Y^\top (K_{\theta_y}(W) + g\mathbb{I}_n)^{-1} Y}{n} \tag{2.4}$$

Throughout, relationship "$\propto$" for log likelihoods denotes that an additive constant has been dropped. Taking $W \equiv X$ lends explicit form to Eq. (1.3) for the ordinary GP case, where $\hat{\tau}^2$ may be interpreted as an MLE. For DGPs with two and three layers, I additionally need

$$\begin{aligned} \log \mathcal{L}(W \mid Z, \theta_w) &\propto \sum_{i=1}^{p} \log \mathcal{L}\left(W_i \mid Z, \theta_w[i]\right) \\ &\propto \sum_{i=1}^{p} \left( -\frac{1}{2} \log |K_{\theta_w[i]}(Z)| - \frac{1}{2} W_i^\top \left(K_{\theta_w[i]}(Z)\right)^{-1} W_i \right). \end{aligned} \tag{2.5}$$

In two-layers, take $Z \equiv X$ and collect

$$\log \mathcal{L}(Y \mid W, X, \theta, g) = \log \mathcal{L}(Y \mid W, \theta_y, g) + \log \mathcal{L}(W \mid X, \theta_w),$$

combining Eqs. (2.4–2.5). Of course, in the context above of unknown (latent) $W$, the quantity $\log \mathcal{L}(W \mid X, \theta_w)$ is actually a log prior, but its form is nevertheless given by Eq. (2.5), so I find it convenient to notate using marginal log likelihoods.

Finally, in the three layer case (with trivial extension beyond), I have

$$\log \mathcal{L}(Y \mid W, Z, X, \theta, g) = \log \mathcal{L}(Y \mid W, \theta_y, g) + \log \mathcal{L}(W \mid Z, \theta_w) + \log \mathcal{L}(Z \mid X, \theta_z)$$

where the third term (and beyond) follows Eq. (2.5) for $Z$ and $X$ instead of $W$ and $Z$. The posterior distribution is completed with the hyperparameter priors outlined at the end of Section 2.1,

$$\pi(W, Z, \theta, g \mid D_n) \propto \mathcal{L}(Y \mid W, Z, X, \theta, g) \times \pi(\theta, g).$$

The remainder of this section details how I obtain samples in the Gibbs framework summarized in Algorithm 1. Hyperparameters $\theta$ and $g$ follow a somewhat conventional random-walk Metropolis-Hastings (MH) procedure. To acknowledge a restricted support $[\varepsilon, \infty]$, proposals ($q$) follow a "uniform sliding window" scheme first outlined by Gramacy and Lee [44] whereby

$$\theta^\star \sim \text{Unif}\left(\frac{\ell\theta^{(t-1)}}{u}, \frac{u\theta^{(t-1)}}{\ell}\right) \quad \text{so that} \quad \frac{q(\theta^{(t-1)} \mid \theta^\star)}{q(\theta^\star \mid \theta^{(t-1)})} = \frac{\theta^{(t-1)}}{\theta^\star} \tag{2.6}$$

features as the proposal ratio in the MH acceptance probability. I sample all components of $\theta$ and $g$ (except when $g \to \varepsilon$ for interpolating deterministic simulations) in this way. Defaults for tuning parameters $u$ and $\ell$ are given in Section 2.4. Under my template, each hyperparameter is involved in only one likelihood component. Thus each MH acceptance

ratio requires only one MVN density evaluation. For example, acceptance for $\theta_y^\star$ is based on

$$\alpha = \min\left(1, \frac{\mathcal{L}\left(Y \mid W, \theta_y^\star, g\right)\pi\left(\theta_y^\star\right)}{\mathcal{L}\left(Y \mid W, \theta_y^{(t-1)}, g\right)\pi\left(\theta_y^{(t-1)}\right)} \times \frac{\theta_y^{(t-1)}}{\theta_y^\star}\right). \tag{2.7}$$

The likelihood component involved in each Metropolis-within-Gibbs step is detailed in Algorithm 1. To shorten this algorithm for a two-layer model, simply remove sampling of $\theta_z$ and $Z$ and replace remaining instances of $Z$ with $X$.

**Algorithm 1:** Gibbs sampling procedure for three-layer DGP

initialize $g^{(1)}$, $\theta_y^{(1)}, \theta_w^{(1)}$, $\theta_z^{(1)}$, $W^{(1)}$,and $Z^{(1)}$

**for** $t = 2, \ldots, T$ **do**

- $g^{(t)} \sim \pi(g \mid Y, W^{(t-1)}, \theta_y^{(t-1)}, g^{(t-1)})$ `// MH (2.7) via` $\mathcal{L}(Y \mid W, \theta_y\ g)$
- $\theta_y^{(t)} \sim \pi(\theta_y \mid Y, W^{(t-1)}, \theta_y^{(t-1)}, g^{(t)})$ `// MH (2.7) via` $\mathcal{L}(Y \mid W, \theta_y\ g)$
- **for** $i = 1, \ldots, p$ **do**
  - $\theta_w[i]^{(t)} \sim \pi(\theta_w[i] \mid W_i^{(t-1)}, Z^{(t-1)}, \theta_w[i]^{(t-1)})$ `// MH (2.7) via` $\mathcal{L}(W_i \mid Z, \theta_w[i])$
- **for** $i = 1, \ldots, p$ **do**
  - $\theta_z[i]^{(t)} \sim \pi(\theta_z[i] \mid Z_i^{(t-1)}, X, \theta_z[i]^{(t-1)})$ `// MH (2.7) via` $\mathcal{L}(Z_i \mid X, \theta_z[i])$
- **for** $i = 1, \ldots, p$ **do**
  - $W_i^{(t)} \sim \pi(W \mid Y, W_{\geq i}^{(t-1)}, W_{<i}^{(t)}, Z^{(t-1)}, g^{(t)}, \theta_y^{(t)}, \theta_w^{(t)})$ `// ESS via (2.9)`
- **for** $i = 1, \ldots, p$ **do**
  - $Z_i^{(t)} \sim \pi(Z \mid X, W^{(t)}, Z_{\geq i}^{(t-1)}, Z_{<i}^{(t)}, \theta_w^{(t)}, \theta_z^{(t)})$ `// ESS via (2.10)`

Latent $W$ and $Z$ could be handled similarly, i.e., by MH. While there are a number of approaches for sampling Gaussian fields in non-conjugate settings [e.g., 54, 92, 95], such rejection-based samplers may mix poorly even with tedious fine tuning of proposal functions. I think this is why MCMC inference for DGPs is often dismissed in preference for

maximization-based shortcuts. However, I find elliptical slice sampling [ESS; 91] of $W$ and $Z$ to be particularly efficient.

To adapt ESS for DGP latent layers, allow me to briefly digress to the general case in which I desire samples from a posterior under a zero-mean MVN prior, $\pi(f) \propto \mathcal{L}(f) \times \mathcal{N}(f; 0, \Sigma)$. ESS initializes with a random draw from the prior $f^{\text{prior}} \sim \mathcal{N}(0, \Sigma)$ and a random angle $\gamma \sim \text{Unif}(0, 2\pi)$ with consequent boundaries set to $\gamma_{\min} = \gamma - 2\pi$ and $\gamma_{\max} = \gamma$. Proposal $f^\star$ is a function of the previous iteration, the prior draw, and the angle,

$$f^\star = f^{(t-1)} \cos\gamma + f^{\text{prior}} \sin\gamma \quad \text{with acceptance probability} \quad \alpha = \min\left(1, \frac{\mathcal{L}(f^\star)}{\mathcal{L}(f^{(t-1)})}\right). \tag{2.8}$$

Upon rejection, ESS shrinks the "bracket" on the angle – specifically, setting $\gamma_{\min} = \gamma$ if $\gamma < 0$ and $\gamma_{\max} = \gamma$ otherwise – and re-samples $\gamma \sim \text{Unif}(\gamma_{\min}, \gamma_{\max})$, yielding new $f^\star$. The process is repeated, shrinking the bracket and re-proposing, until acceptance, constituting one iteration/one sample. This distinguishes ESS from MH methods in which rejections result in identically repeated values and poor mixing. Although ESS may require more likelihood evaluations (each new proposal warrants one) than MH, it tends to yield more "effective samples" [63] through enhanced mixing, as demonstrated in Section 2.2, all without having to set any tuning parameters.

To implement ESS for $W$ and $Z$, one may leverage the likelihood (2.5) as a prior over the unknown function. First generate from the prior for each node,

$$W_i^{\text{prior}} \sim \mathcal{N}_n\left(0, K_{\theta_{w[i]}}(Z)\right) \quad \text{and} \quad Z_i^{\text{prior}} \sim \mathcal{N}_n\left(0, K_{\theta_{z[i]}}(X)\right).$$

In the two-layer case, set $Z \equiv X$ for each $W_i$. ESS proposals $W_i^\star$ and $Z_i^\star$ follow Eq. (2.8) based on previous iteration $W_i^{(t-1)}$ and $Z_i^{(t-1)}$ respectively. Acceptance probabilities are based on a likelihood component designed to "receive" that latent quantity as an input.

Even though nodes of a layer are independently proposed, all are involved in the model likelihood (under imposition iii.) and hence impact acceptance. For $W_i$, in either a two- or three-layer model, that involves Eq. (2.4),

$$\alpha = \min\left(1, \frac{\mathcal{L}(Y \mid W_i^\star, W_{>i}^{(t-1)}, W_{<i}^{(t)}, \theta_y, g)}{\mathcal{L}(Y \mid W_{\geq i}^{(t-1)}, W_{<i}^{(t)}, \theta_y, g)}\right) \tag{2.9}$$

where, in true Gibbs fashion, I condition on the most recent iteration of each node. Acceptance probabilities for $Z_i$ use the likelihood of Eq. (2.5).

$$\alpha = \min\left(1, \frac{\mathcal{L}(W \mid Z_i^\star, Z_{>i}^{(t-1)}, Z_{<i}^{(t)}, \theta_w)}{\mathcal{L}(W \mid Z_{\geq i}^{(t-1)}, Z_{<i}^{(t)}, \theta_w)}\right) \tag{2.10}$$

In contrast to $\theta_w[i]$, which only features in one component of this likelihood, acceptance of node $Z_i$ requires full evaluation of this likelihood (including all $p$ elements of the sum). Specific implementation details are reserved for Section 2.4.

### Illustration

MCMC sampling of latent layers $W$ and $Z$ is crucial to downstream tasks, such as AL in Chapter 3. Their posteriors are inherently multi-modal, since only pairwise distances feature in the next layer. Such lack of identifiability poses a challenge to MCMC mixing and posterior exploration, but on the flip side of that coin is a convenient barometer for judging a proposed sampling scheme. My ESS proposal mechanism (2.8) is particularly well-suited for this as it can "bounce" back and forth between modes. To demonstrate, consider the setup in Figure 1.2 with a two-layer DGP. The fitted surface is shown in the right panel, but I have not yet discussed prediction (Section 2.3). For now, focus on latent $W$ via ESS shown in the left panel of Figure 2.3. Notice how $W$ samples bend inputs $X$ so that the middle inputs are distinguished from extremities, separating flat interior dynamics from outer wiggly ones

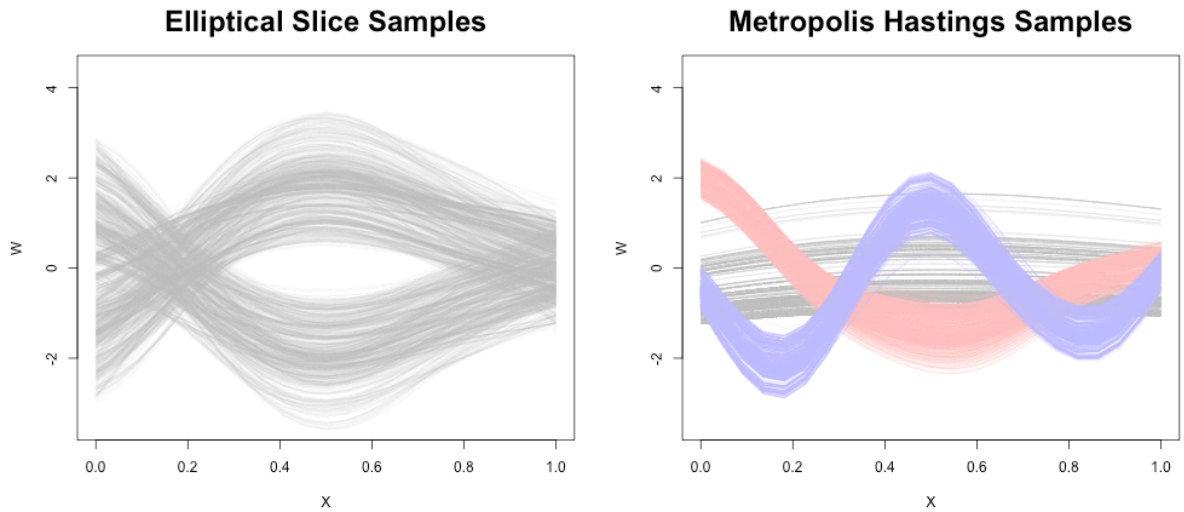


Figure 2.3: ESS and MH samples of latent $W$ from the two-layer DGP of Figure 1.2, otherwise following Algorithm 1 for hyperparameters $\theta$ and $g$. MH samples show three different initializations; each (three MH & one ESS) comprise 10,000 total and 6,000 burn-in iterations with thinning by half.

either by bending down then back up or up then back down.

To offer contrast, I implemented an MH alternative for sampling $W$ using kernel-based basis proposals [similar to 54, 114] conditioned on sampled $\theta_w^{(t)}$. See the right panel in Figure 2.3. MH samples were highly sensitive to initialization and step size. Depending on how the chain is initialized, with three random settings shown (and indicated with color), I get very different $W$, a telltale sign of poor mixing/convergence of the chain. In each case shown, the step size was finely tuned to achieve an acceptance rate of 30-35%. This tuning and re-initialization requires many more MCMC iterations than the ESS comparator and makes burn-in of the chain difficult to assess. Posterior predictions calculated from these MH samples (following Section 2.3, but not shown) all resulted in poorer out-of-sample root mean square errors (RMSEs) than those fit via ESS under similar computational budgets.

## 2.3 Posterior Prediction

Here I extend MCMC sampling (Section 2.2) to posterior predictive draws $Y(\mathcal{X}) \mid D_n$ at $n' \times d$ testing locations specified row-wise in $\mathcal{X}$. For simplicity I limit the discussion here to the two-layer DGP case. Extending to three layers is straightforward, albeit cumbersome notationally.

After conducting MCMC sampling and any thinning or burn-in (details in Section 2.4), I am left with samples $\{\hat{\tau}^{2(t)}, g^{(t)}, \theta_w^{(t)}, \theta_y^{(t)}, W^{(t)} \mid t \in \mathcal{T}\}$ from the posterior distribution of all unknowns. Although all of these are involved, development of the relevant predictive quantities hinges most crucially on $W$. Recall that $W$ follows a GP in $X$, and $Y$ a GP in $W$. Samples $W^{(t)}$ in hand, I may utilize predictive equations (1.4) to calculate posterior moments over $\mathcal{X}$. Combining with Eq. (2.3) yields that $\mathcal{W} \equiv W(\mathcal{X})$ is MVN. For a particular sample indexed by $t$, the moments are:

$$\begin{aligned}
\mu_{W_i}^{(t)}(\mathcal{X}) &= K_{\theta_w^{(t)}[i]}(\mathcal{X}, X)(K_{\theta_w^{(t)}[i]}(X))^{-1} W_i^{(t)} \\
\Sigma_{W_i}^{(t)}(\mathcal{X}) &= K_{\theta_w^{(t)}[i]}(\mathcal{X}) - K_{\theta_w^{(t)}[i]}(\mathcal{X}, X)(K_{\theta_w^{(t)}[i]}(X))^{-1} K_{\theta_w^{(t)}[i]}(X, \mathcal{X}).
\end{aligned} \tag{2.11}$$

Sampled $\mathcal{W}^{(t)}$, whose values represent a warping of predictive locations $\mathcal{X}$, may then be fed into the outer-layer to derive posterior moments for $\mathcal{Y} \equiv Y(\mathcal{X})$. Together with $\hat{\tau}^{2(t)}$ following Eq. (2.4) evaluated with $W^{(t)}$ and $\theta_w^{(t)}$, similar arguments as for $\mathcal{W}$ yield that $\mathcal{Y}^{(t)}$ follows a multivariate Student-$t$ predictive distribution with $n$ degrees of freedom and

$$\begin{aligned}
\mu_Y^{(t)}(\mathcal{W}^{(t)}) &= K_{\theta_y^{(t)}}(\mathcal{W}^{(t)}, W^{(t)})(K_{\theta_y^{(t)}}(W^{(t)}) + g^{(t)}\mathbb{I}_n)^{-1} Y \\
\Sigma_Y^{(t)}(\mathcal{W}^{(t)}) &= \hat{\tau}^{2(t)} \left[ K_{\theta_y^{(t)}}(\mathcal{W}^{(t)}) + g^{(t)}\mathbb{I}_{n'} - K_{\theta_y^{(t)}}(\mathcal{W}^{(t)}, W^{(t)})(K_{\theta_y^{(t)}}(W^{(t)}) + g^{(t)}\mathbb{I}_n)^{-1} K_{\theta_y^{(t)}}(W^{(t)}, \mathcal{W}^{(t)}) \right].
\end{aligned} \tag{2.12}$$

Instead sampling $\tau^{2(t)}$ from its conjugate posterior conditional $\mathrm{IG}(\frac{n}{2}, \frac{n\hat{\tau}^2}{2})$, and swapping in

for $\hat{\tau}^{2(t)}$ would be MVN. In practice I find that this distinction hardly matters, and it is sufficient to treat the former, in spite of estimated scale $\hat{\tau}^{2(t)}$, as MVN except when $n$ is very small. Although posterior predictive uncertainty is most completely described by a large collection of $\mathcal{Y}^{(t)}$, it can be far more compact to accumulate moments through the law of total expectation and variance. As long as the samples $\mathcal{T}$ retained number more than $n'$,

$$\mu_Y = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \mu_Y^{(t)} \quad \text{and} \quad \Sigma_Y = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \Sigma_Y^{(t)} + \frac{1}{|\mathcal{T}|-n'}\sum_{t\in\mathcal{T}} (\mu_Y^{(t)} - \mu_Y)(\mu_Y^{(t)} - \mu_Y)^\top, \tag{2.13}$$

are moments depicting an MVN approximation to the empirical distribution $\{\mathcal{Y}^{(t)}\}_{t\in\mathcal{T}}$.

There are a few other shortcuts I find handy. Unless sample paths of $Y(\mathcal{X})$ are desired, one may short-cut a full covariance calculation – i.e., $\Sigma_Y^{(t)}(\mathcal{W}^{(t)})$ in Eq. (2.12) – for its diagonal (variance) components yielding point-wise equations described by independent scalar Gaussian (or Student-$t$) equations. This saves on both storage and computational effort, with the latter being quadratic rather than cubic on $n'$. Variances are sufficient for evaluating AL criteria in Chapter 3 and for calculating quantile-based error-bars. Such quantities would still capture all relevant uncertainties for most purposes, e.g., for posterior predictive intervals. It may moreover be sufficient, and even desirable, to describe mean uncertainty only, i.e., for $\mathbb{E}\{Y(\mathcal{X})\}$ rather than $Y(\mathcal{X})$. Simply replace $g^{(t)}\mathbb{I}_{n'}$ with the zero matrix in Eq. (2.12). This is common in AL applications (Chapter 3) and when building "confidence intervals" on prediction. Note that the nugget is still involved in $(K_{\theta_y^{(t)}}(W^{(t)}) + g^{(t)}\mathbb{I}_n)^{-1}$, which is what induces smoothing.

Even after reducing to point-wise calculations in $Y(\mathcal{W}^{(t)})$, extracting variances instead of covariances, the entire process is still cubic in $n'$ because sampling $\mathcal{W}^{(t)}$ involves $n' \times n'$ matrices (2.11), which could be prohibitive. However, I often find it sufficient to skip calculating $\Sigma_{W_i}^{(t)}(\mathcal{X})$, and subsequent MVN draws, and instead take $\mathcal{W}_i^{(t)} \equiv \mu_{W_i}^{(t)}(\mathcal{X})$. This results in an

underestimate of predictive uncertainty, but not substantially so and not in a way that has any detectable effect on downstream tasks. In the running example of Figure 1.2, sampled $\mathcal{W}^{(t)}$ (as plotted in Figure 2.3) are indistinguishable from $\mu_W^{(t)}(\mathcal{X})$ and produce equivalent predictions.

## 2.4 Implementation Details

All empirical work in this manuscript is supported by the `deepgp` package on CRAN [110]. Defaults, described momentarily, are used throughout except where noted. [For example, I entertain $p < d$ in Section 3.1.5.] These defaults are appropriate for inputs $X$ coded to $[0,1]^d$ and outputs $Y$ pre-processed to have mean zero and variance one. While the ESS component of Algorithm 1 is free of tuning parameters, hyperparameter priors and proposals are required for $\theta$ and $g$. In the uniform sliding window proposal of Eq. (2.6), I use $l = 1$ and $u = 2$ for $g$ and $\theta$ at all layers.

Recall from Section 2.1 that I take $\{\theta, g\} \overset{\text{iid}}{\sim} \text{G}(3/2, b_{[\cdot]})$. For $g$ I select a rate so that 95% of the prior mass lies below 1 (the scaled data variance), resulting in $b_{[g]} = 3.9$. Lengthscales $\theta$ proceed similarly, but I differentiate between those which act on coded inputs $X$ spanning $[0,1]$, those for latent layers under MVN spread, i.e., spanning $[-2,2]$ at 95% (see Figure 2.3), and latent depth. In a three-layer model, $\theta_z$ acts on $X$, and I want to encourage an identity latent mapping in $Z$. This helps regularize the system in the face of dynamics which are stationary, nearly so, or for which I do not yet have enough data (say in an AL context) to identify regime shifts. I thus choose $b_{[\theta_z]} = 3.9/4$, so that 95% of the prior mass falls at less than four times the maximum pairwise distances in $X$. In a two-layer model, where $\theta_w$ acts on $X$, I choose $b_{[\theta_w]}$ similarly. The outer layer always acts on latent inputs in $[-2,2]$, but I am willing to substantially diverge from identity mappings – after

all, presumably one entertains GPs because of an *a priori* belief in nonlinearity. Following similar logic, but this time targeting 95% prior mass at less than 1.5 times the maximum pairwise distance in $[-2, 2]$, I choose $b_{[\theta_y]} = 3.9/6$. This choice is mirrored in one-layer models, setting $b_{[\theta]} = 3.9/1.5$ for an isotropic lengthscale acting on coded $X$. Finally, in three-layer models I preserve this hierarchy with $b_{[\theta_w]} = 3.9/12$ (95% mass less than 3 times the maximum pairwise distance). These settings are, of course, all user-adjustable.

Conducting MCMC sampling for two- and three-layer models in the `deepgp` package using built-in defaults is as simple as the following, producing distinct S3-class objects.

```
R> fit.2 <- fit_two_layer(X, Y)
R> fit.3 <- fit_three_layer(X, Y)
```

Unless otherwise specified, the dimension of hidden layers will match that of inputs, i.e., $p = d$. For lower dimensional problems, it is important to allow for this flexibility, if only to imitate the kernel structure of separable processes. Smaller $p < d$ can have deleterious effects except when the majority of (an already small number of) inputs are irrelevant, which is rare in computer surrogate modeling applications. In higher dimensional settings, upwards of $d = 7$ as in Sections 3.1.5, some tension between the dimension of inputs and layers can have a beneficial "autoencoding effect", uncovering a lower-dimensional latent structure such as may arise when dynamics unfold on a sub-manifold of inputs.

Once samples are collected, one may remove early iterations, saving burned-in samples indexed as $t \in \mathcal{T}$. To reduce the size of this set, samples may further be thinned. Both are facilitated by an S3 method called `trim`.

After MCMC, which is an inherently serial affair, all downstream tasks are parallelizable across iterations $t \in \mathcal{T}$. Predictions may be made separately for each $t$, with expectations (2.13) calculated afterwards in a map-reduce fashion. When only point-wise means and

variances are required, predictions may be parallelized across testing locations, $\mathcal{X}$. Parallel implementations of prediction are provided by the S3 method `predict` via `foreach` [88] constructs.

## 2.5 Synthetic Experiment

I will provide thorough empirical results in Chapters 3–4, but for now I will preview one of the examples explored later in Section 4.4. Consider the G-function [83], defined in arbitrary dimension on the pages of the Virtual Library of Simulation Experiments [124],

$$f(x) = \prod_{i=1}^{d} \frac{|4x_i - 2| + a_i}{1 + a_i} \quad \text{where} \quad a_i = \frac{i-2}{2} \quad \text{for} \quad i = 1, \ldots, d. \tag{2.14}$$

A visual in two dimensions is provided in the left panel of Figure 2.4. This function is characterized by steep inclines and abrupt shifts. A stationary "shallow" GP is unable to distinguish between the steep sloping regions in the corners and the valley regions that form a cross shape in the center. Figure 2.4 also shows a posterior sample (i.e., a burned-in ESS draw) of the hidden layer $W$ plotted as a function of $X$ with each of two "nodes" in their own pane. These nodes act together to stretch the inputs in each diagonal direction – thus accommodating the diagonally-oriented steep inclines seen on the left.

Two dimensions provide a clear visual, but they are not very interesting from a surrogate modeling perspective. For a more realistic test, I consider four dimensions. I fit both one-layer GPs and two-layer DGPs to equivalent training data sets of various sizes. I evaluated the predictions using root mean square error (RMSE, providing a measure of accuracy of mean predictions) and continuous rank probability score [CRPS; 39, providing a measure of uncertainty quantification]. For both metrics, lower values indicate better fits. Figure

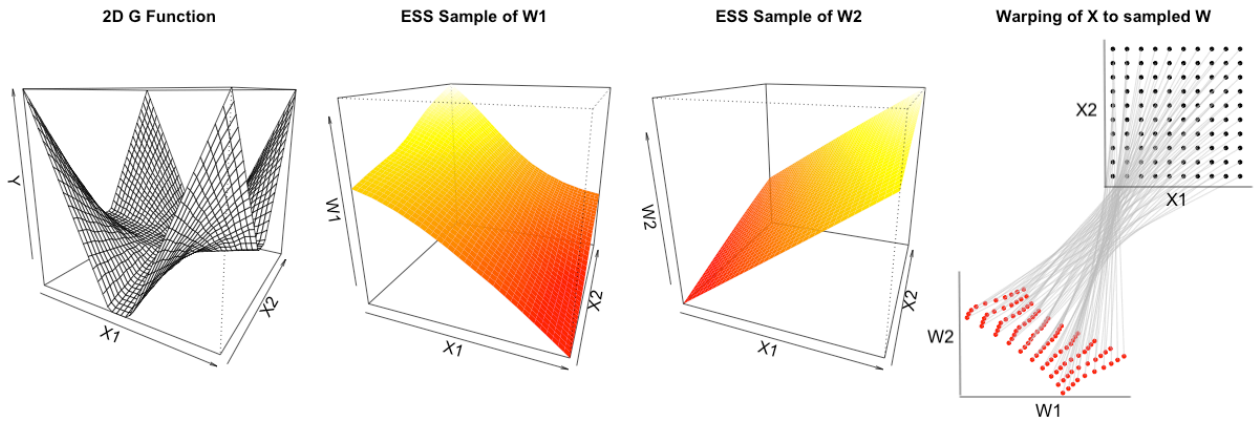


Figure 2.4: (Left) Visual of the two-dimensional G-function. (Middle) Posterior sample of $W$ for a two-layer DGP fit to the data of the left panel, comprised of two "nodes" (one in each panel). (Right) The same ESS sample, now visualized jointly as a warping of $X$.

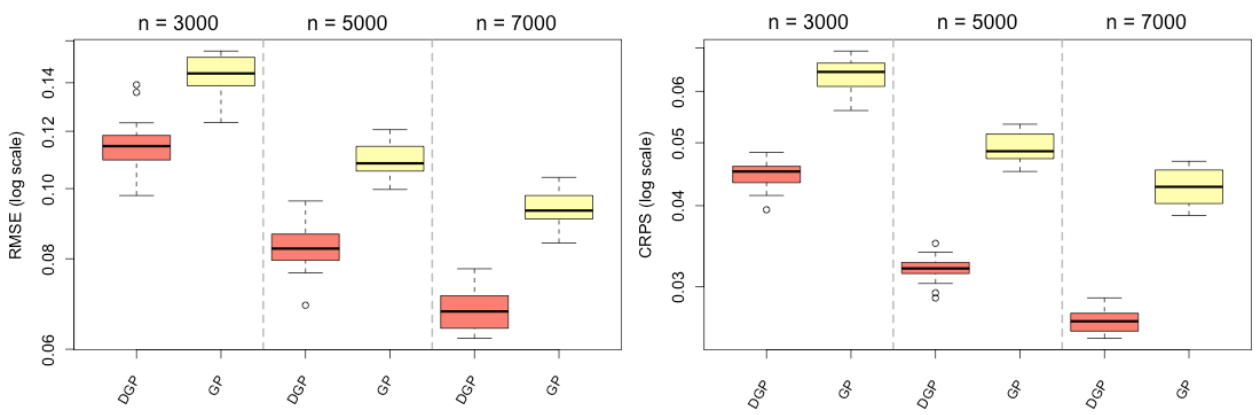


Figure 2.5: RMSE (left) and CRPS (right) on log scales for fits to the four-dimensional G-function as training size ($n$) increases. Boxplots represent the spread of 10 MC repetitions.

2.5 displays results across 10 Monte Carlo (MC) repetitions. The flexibility of the DGP model results in better predictions (in terms of both accuracy and UQ). This example will be featured and expanded upon in Section 4.4; here I use it as merely a precursor of potential.

## 2.6 Discussion

I provided a novel Gibbs-ESS-Metropolis method for fitting deep Gaussian processes, specifically targeted towards surrogate modeling applications. In addition to illustrations of component parts of my inferential scheme, I demonstrated a preview of DGP superiority over typical GP regression on a synthetic example. I reserve further empirical evidence of DGP performance for later chapters.

The recent success of DGPs as surrogates, such as for calibration [81], multi-fidelity modeling [19], and black-box optimization [53], is exciting. UQ plays a key role in each of these, yet approximation via maximization is the *modus operandi*. Consequently, such surrogates under-estimate – or at best mis-characterize – relevant uncertainties. My novel Gibbs-ESS-Metropolis scheme makes MCMC tractable where it wasn't previously effective.

I made several modeling choices in order to protect identifiability and to favor simplicity. These were motivated by challenges brought to light in empirical work behind earlier incantations. Examples include enforcing common dimensionality in latent layers, limiting depth to three-layers (or even two), and limiting nodes to no more than the input dimension. These constraints seem to provide a degree of regularization which does not overly hinder reactivity of the apparatus. However, a more detailed cost-benefit analysis of DGP complexity v. prowess may be warranted.

The additional computation required to fit DGP models poses a challenge for larger sample sizes. My MCMC method in its current state is not appropriate when data is abundant. I explore one remedy, the Vecchia approximation, in Chapter 4.

# Chapter 3

# Active Learning for DGPs

The most common designs for computer simulation experiments are based on space-filling principles, either along the margins with Latin hypercube samples [LHS; 85], via pairwise distances [59], or hybrids thereof [90]. These "static" strategies presume a fixed budget of $n$ runs in advance. Sequential adaptations exist but do not leverage model uncertainties. For a review, see Gramacy [42, Chapter 4]. Model-based analogues, which devise criteria via prior-posterior information gain [so-called maximum entropy designs; 116] and posterior mean-squared prediction error (MSE), offer similar behavior. But these have a chicken-or-egg problem because they condition on (*a priori* unknown) hyperparameter settings. [Guessing good hyperparameters at the start is even less feasible for a DGP with its unknown latent layers and layered kernels.] Sequential adaptations that leverage and update model information along the way offer a natural remedy, especially when suitably initialized [132]. See Gramacy [42, Chapter 6] for review.

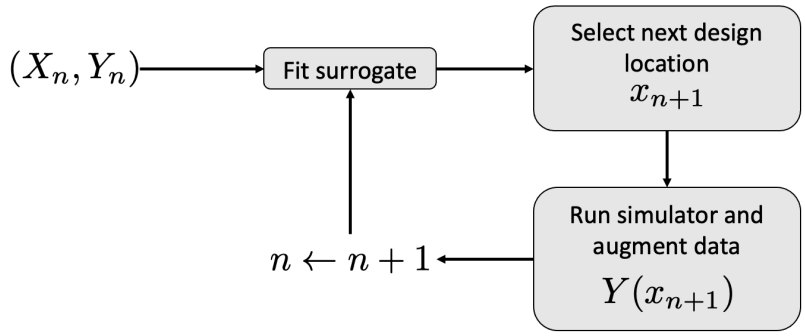


Figure 3.1: Illustration of an active learning procedure.

Much recent work on sequential design for GPs comes from machine learning (ML) as a branch of reinforcement learning called active learning (AL). I adopt their terminology of solving acquisition functions (sequential design criteria) to "actively" select the data which the model is trained on. Figure 3.1 provides an illustration of this process. Begin (0) with a small training data set $D_n = \{X_n, Y_n\}$ of size $n = n_0$; then (1) fit a flexible surrogate to $D_n$; (2) select $x_{n+1}$ to augment the data $n \to n+1$; and (3) repeat from (1). This setup avoids many pathologies inherent to static/batch training through a baby-steps approach and allows progress to be monitored. The "acquisition step" (step 2) may be chosen to target specific design objectives. Acquisitions based on aggregate variance reduction [15] aim to minimize predictive error. Acquisitions based on expected improvement target optima in blackbox simulation output [e.g., 46, 98, 118]. Acquisitions based on entropy (or hybridizations thereof) target level sets in the respone surface [e.g., 16, 119]. Sequential designs aimed at calibrating computer simulations to real-world data [70] are a newer development [72]; I will defer DGP-based active learning for simulator calibration to future research.

The novelty of my work here stems from upgrading the surrogate model (step 1) to a deep Gaussian process. This upgraded surrogate makes evaluation of acquisition criteria (step 2) more challenging, necessitating further advancements. Note, the UQ provided by the Bayesian MCMC scheme outlined in Chapter 2 will be especially impactful, as surrogate-based variance estimates feature heavily in each acquisition criterion.

## 3.1 Variance Reduction

Here I emphasize building up a simulation campaign where runs are acquired to reduce MSE. Let $X_{n+1} = \{x_1, \ldots, x_n, x_{n+1}\}$ represent the combined set of $n$ current input locations and new input location $x_{n+1}$. Let $\sigma_n^2(x)$ denote the posterior predictive variance (i.e., the

MSE) at location $x$ calculated from $X_n$, i.e., via Eq. (1.4) with singleton $\mathcal{X} = \{x\}$. Take hyperparameter settings from maximum likelihood estimates (MLE) or as posterior samples conditioned on data $D_n$. Now, let $\breve{\sigma}^2_{n+1}(x)$ denote the deduced posterior predictive variance at location $x$ calculated from the augmented $X_{n+1}$, but otherwise with hyperparameter settings based on $D_n$.

$$\breve{\sigma}^2_{n+1}(x) = \Sigma(x) - \Sigma(x, X_{n+1})\Sigma^{-1}_{n+1}\Sigma(X_{n+1}, x)$$

I follow the convention of defining both $\sigma^2_n(x)$ and $\breve{\sigma}^2_{n+1}(x)$ for the purpose of AL via $\Sigma(x) = 1$ rather than $\Sigma(x) = 1 + g$, targeting the variance of the latent random field (i.e., the variance of the mean) rather than the full predictive variance. Making this, and other kernel details explicit,

$$\breve{\sigma}^2_{n+1}(x) = \tau^2 \left(1 - K_\theta(x, X_{n+1}) \left(K_\theta(X_{n+1}) + g\mathbb{I}_{n+1}\right)^{-1} K_\theta(X_{n+1}, x)\right).$$

Hence $\breve{\sigma}^2_{n+1}(x) \to 0$ as $n \to \infty$, providing a natural "asymptote for learning." Observe that the nugget $g$ is still involved in $\Sigma^{-1}_{n+1}$.

It is sensible to acquire new $x^\star_{n+1}$ to minimize $\breve{\sigma}^2_{n+1}(x)$ integrated over all $x$,

$$x^\star_{n+1} = \underset{x_{n+1}}{\operatorname{argmin}}\ \mathrm{IMSE}(X_{n+1}) \quad \text{where} \quad \mathrm{IMSE}(X_{n+1}) = \int \breve{\sigma}^2_{n+1}(x)\, dx.$$

When the input space used in the domain of integration is a hyper-rectangle, a closed form expression is available. See, e.g., Binois et al. [11] for coded inputs $[0, 1]^d$ and extensions by Cole et al. [16] to $[a, b]^d$ and Gaussian measures over inputs.

Despite a degree of analytic tractability, calculating IMSE is cumbersome, especially in the Bayesian posterior sampling context. Varying hyperparameterization thwarts many pre-calculations and matrix decomposition tricks that would otherwise lend a degree of efficiency.

Instead I prefer an approximation obtained by extending Monte Carlo (MC) integration over posterior draws to include the input space. This is the acquisition criteria now known as "active learning Cohn" [ALC; 15, 115]. Observe that minimizing IMSE over $x_{n+1}$ is equivalent to maximizing the difference between $\sigma_n^2(x)$ and $\breve{\sigma}_{n+1}^2(x)$.

$$\Delta\sigma_n^2(X_{n+1}) = \int \sigma_n^2(x) - \breve{\sigma}_{n+1}^2(x)dx \propto \int \tau^2 K_\theta(x, X_{n+1}) \left(K_\theta(X_{n+1}) + g\mathbb{I}_{n+1}\right)^{-1} K_\theta(X_{n+1}, x)dx$$

The ALC criterion approximates this integral with a sum over a reference set, $X_{\text{ref}}$,

$$\text{ALC}(X_{n+1} \mid X_{\text{ref}}) \propto \sum_{x \in X_{\text{ref}}} \tau^2 K_\theta(x, X_{n+1}) \left(K_\theta(X_{n+1}) + g\mathbb{I}_{n+1}\right)^{-1} K_\theta(X_{n+1}, x) \tag{3.1}$$

with acquisition as $x_{n+1}^\star = \text{argmax}_{x_{n+1}} \text{ALC}(X_{n+1} \mid X_{\text{ref}})$. A uniform reference set yields proportional ALC and IMSE surfaces. See left and center panels of Figure 1.2. Instead of library-based optimization [43], I prefer discrete search over (possibly random) candidates because this lends well to averaging across MCMC iterations and parallel implementation.

Even with variance-based criteria, AL is limited by the stationarity of the covariance kernel of the GP [45]. In Figure 1.2, the MLE surface (left) gets the "wrong answer" as regards AL, discouraging acquisition where it needs it most: in the middle of inputs where it has confused noise as signal. With MCMC (center), the ALC/IMSE criteria indentify a more uniform preference for the next run, although both ultimately prefer runs near the edges, which manifests as a form of leverage in this case. Neither MLE nor MCMC-based fits are able to recognize what human intuition would likely suggest: more data is needed where the function is wiggly than where it is not. This is not the fault of the AL criteria. The stationary GP is not "allowed" to recognize two (or more) distinct regimes, of wiggly and flat. Only after relaxing stationarity, so that predictive variance is not simply a matter of distance to nearby training values, can we obtain a notion of predictive uncertainty which is

region-dependent.

For another illustration, consider a typical, stationary GP surrogate fit to simulations following Eq. (3.5), detailed in Section 3.1.3. The left panel of Figure 3.2 shows ALC, using a dense $\mathcal{X}_{\text{ref}}$ grid, after training on evaluations obtained from an LHS of size $n = 30$. In this example, the lower left corner is the area of highest interest, but ALC is, in essence, a function of proximity to $X_n$. A two-layer DGP fit to the same training data is able to allocate uncertainty in the "interesting" region, resulting in non-space filling acquisitions.

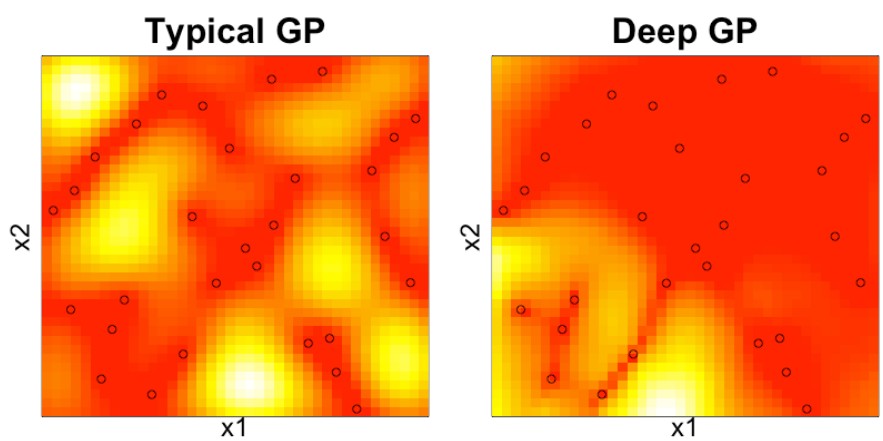


Figure 3.2: Heat map of ALC surface for typical GP and two-layer DGP surrogates (via MCMC) for $f(x_1, x_2)$ in Eq. (3.5). Open circles indicate design $X_n$. White/high, red/low.

### 3.1.1 Acquisition

Given training data $D_n = \{X_n, Y_n\}$, AL seeks to optimize a criterion conceived to assess the potential value of $x_{n+1} \mid D_n$ at which to acquire a new $y_{n+1} = f(x_{n+1})$, ultimately forming augmented data $D_{n+1} = \{X_{n+1}, Y_{n+1}\}$ with $x_{n+1}^\top$ and $y_{n+1}$ in the last row and entry, respectively. The heftiness of the MCMC-based DGP is at odds with numerical optimizers [e.g., 76] which require repeated successive evaluation of the simulator at an *a priori* unknown number of locations. I instead embrace discrete optimization through

candidate evaluation. All candidate locations may be evaluated simultaneously, possibly in parallel, which assuages computational costs. The pre-set size of the candidate set also offers control over computational expenses.

Let $\mathcal{X}$ denote a set of candidates from which $x_{n+1}$ will be selected. Criterion in hand, AL acquisition with DGPs may proceed as follows: (1) collect MCMC samples indexed by $t \in \mathcal{T}$ (Section 2.2), (2) map $\mathcal{X}$ to the criterion for each $t$; (3) average over $t \in \mathcal{T}$; (4) choose from $\mathcal{X}$ optimizing the average. The relevant calculations boil down to propagating testing inputs $\mathcal{X}$ (i.e., the AL candidates) through latent layers $\mathcal{X} \to \mathcal{W} \to \mathcal{Y}$ over MCMC iterations $t \in \mathcal{T}$.

Let $W_{n+1}^{(t)}$ denote the row-combined set of mapped latent layer values for the existing data $W_n^{(t)}$ and those (transposed values) corresponding to candidate/testing location $x_{n+1} \in \mathcal{X}$, commensurately $w_{n+1}^{(t)} \in \mathcal{W}^{(t)}$, for a particular MCMC iteration $t \in \mathcal{T}$. I find it helpful to denote the un-scaled covariance as $C_{n+1}^{(t)} = K_{\theta_y^{(t)}}(W_{n+1}^{(t)}) + g^{(t)}\mathbb{I}_{n+1}$, where only the last row and column depend on $w_{n+1}^{(t)}$, and $C_n^{(t)}$ similarly. Whereas $\mathcal{W}^{(t)}$ quantities play a fundamental role in the input warping behind the predictive scheme from Section 2.3, $C_{n+1}^{(t)}$ evaluated point-wise for all $w_{n+1} \leftarrow x_{n+1} \in \mathcal{X}$ is fundamental to ALC and IMSE acquisition under DGPs.

Consider IMSE first. Binois et al. [11] and Joseph et al. [61] integrated predictive variance, i.e., $\Sigma_Y^{(t)}(w)$ in Eq. (2.12), in closed form under a uniform measure in the unit cube. This is fine in ordinary GP applications where $\mathcal{X}$ can be trivially coded to $[0,1]^d$, but $w$'s are Gaussian and thus have support on the whole real line (in each of $p$ coordinates). Maintaining integration over a uniform measure, in keeping with the spirit of earlier IMSE applications, thus required an upgrade. Letting $a^{(t)} = \min\{\mathcal{W}^{(t)}\}$, applied column-wise to calculate a $p$-vector, and similarly $b^{(t)} = \max\{\mathcal{W}^{(t)}\}$, I derived the following closed form IMSE over a

uniform measure in $[a^{(t)}, b^{(t)}]$ and Gaussian kernel (1.2)

$$\text{IMSE}(w_{n+1}^{(t)}) = \hat{\tau}^{2(t)} \prod_{i=1}^{p} (b_i^{(t)} - a_i^{(t)}) \left[ 1 - \text{tr}\left( (C_{n+1}^{(t)})^{-1} H^{(t)} \right) \right] \tag{3.2}$$

where $H^{(t)}$ is an $(n+1) \times (n+1)$ matrix with elements

$$H_{jk}^{(t)} = \left( \frac{\pi \theta_y^{(t)}}{2} \right)^{\frac{p}{2}} \prod_{i=1}^{p} \left\{ \exp\left( -\frac{(w_{j,i} - w_{k,i})^2}{2\theta_y^{(t)}} \right) \left[ \Phi\left( \frac{2b_i - w_{j,i} - w_{k,i}}{\sqrt{\theta_y^{(t)}}} \right) - \Phi\left( \frac{2a_i - w_{j,i} - w_{k,i}}{\sqrt{\theta_y^{(t)}}} \right) \right] \right\},$$

in which $w_{j,i}$ is the $j^{th}$ element of the $i^{th}$ node of $W_{n+1}^{(t)}$ and $\Phi$ is the standard Gaussian CDF. A detailed derivation is provided in Appendix A.2. IMSE is a function of $w_{n+1}^{(t)}$, as the first $n$ rows of $W_{n+1}^{(t)}$ are fixed. The first $n \times n$ block of $H^{(t)}$ may be pre-calculated as only the final row and column depend on $w_{n+1}^{(t)}$. Similarly, pre-calculating $(C_n^{(t)})^{-1}$ using partitioned matrix inverse (Appendix A.1), allows for linear computation of $(C_{n+1}^{(t)})^{-1}$ for each $w_{n+1}^{(t)}$.

ALC proceeds similarly, but requires the specification of an additional reference set $\mathcal{X}_{\text{ref}}$. I prefer $\mathcal{X}_{\text{ref}} \equiv \mathcal{X}$, which is the default in other/ordinary GP contexts and simplifies matters considerably. Although one could map novel $\mathcal{X}_{\text{ref}} \to \mathcal{W}_{\text{ref}}^{(t)}$ following Eq. (2.11), it is far easier to take $\mathcal{W}_{\text{ref}}^{(t)} = \mathcal{W}^{(t)}$. I simplify notation by denoting $\mathbf{k}_n^{(t)}(w) = K_{\theta_w^{(t)}}(W_n^{(t)}, w)$ and $\mathbf{k}_{n+1}^{(t)}(w) = K_{\theta_w^{(t)}}(W_{n+1}^{(t)}, w)$. Following Eq. (3.1), the ALC statistic may be developed as follows, with details in Appendix A.3.

$$\begin{aligned} \text{ALC}(W_{n+1}^{(t)} \mid \mathcal{W}_{\text{ref}}^{(t)}) &= \sum_{w \in \mathcal{W}_{\text{ref}}^{(t)}} \hat{\tau}^{2(t)} k_{n+1}^{(t)\top}(w) (C_{n+1}^{(t)})^{-1} k_{n+1}^{(t)}(w) \\ &\propto \sum_{w \in \mathcal{W}_{\text{ref}}^{(t)}} \hat{\tau}^{2(t)} \left[ v(h^\top k_n^{(t)}(w))^2 + 2zh^\top k_n^{(t)}(w) + v^{-1}z^2 \right], \end{aligned} \tag{3.3}$$

with $v = 1 + g^{(t)} - k_n^{(t)\top}(w_{n+1}^{(t)})(C_n^{(t)})^{-1} k_n^{(t)}(w_{n+1}^{(t)})$, $h = -v^{-1}(C_n^{(t)})^{-1} k_n^{(t)}(w_{n+1}^{(t)})$, and $z = K_{\theta_y^{(t)}}(w_{n+1}, w)$.

Aggregate criteria (either IMSE or ALC) for $x_{n+1}$, mapped to $w_{n+1}^{(t)}$ for each $t \in \mathcal{T}$ arise as

$$\text{IMSE}(x_{n+1}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \text{IMSE}(w_{n+1}^{(t)}) \quad \text{and} \quad \text{ALC}(x_{n+1}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \text{ALC}(w_{n+1}^{(t)} \mid \mathcal{W}_{\text{ref}}^{(t)}).$$

Depending on the preferred criterion, acquisition involves selecting $x_{n+1} = \text{argmin}_x \text{IMSE}(x)$ or $x_{n+1} = \text{argmax}_x \text{ALC}(x)$. These acquisitions could be solved using a numerical optimizer, but again I prefer discrete search over a candidate set because it lends well to pre-calculations, averaging over iterations, and parallel implementation.

Extending my earlier illustrations, the right panel of Figure 1.2 shows DGP-based IMSE and ALC evaluated over a dense candidate set $\mathcal{X}$. Both criteria favor acquisitions in the left, more wiggly region. ALC's reference set $\mathcal{X}_{\text{ref}} = \mathcal{X}$, which is uniform over the inputs, becomes warped through the latent $W$ and yields non-uniform $\mathcal{W}_{\text{ref}}$, accounting for the slight differences between the IMSE and ALC surfaces. This discrepancy can become more pronounced in three-layer models, and higher. Figure 3.2 provides an illustration of the ALC surface for the 2d example of Section 3.1.3. When combined with the non-stationary DGP, the ALC criterion is able to identify the area of highest interest (lower left corner).

### 3.1.2 Implementation Details

IMSE and ALC calculations are provided in the `deepgp` package through the S3 methods `IMSE` and `ALC`, which are parallelized via `foreach` [88] constructs. Under-the-hood calculations utilize `C` and `C++` for faster computation. These functions require an S3 class object containing burned-in MCMC samples and the specification of a set of candidates.

```
R> fit.2 <- fit_two_layer(X, Y)
R> imse.2 <- IMSE(fit.2, Xcand)
```

```
R> alc.2 <- ALC(fit.2, Xcand)
```

When designing sequentially with AL, after new $\{x_{n+1}, f(x_{n+1})\}$ is acquired, it is helpful to re-start MCMC samples at the values where the previous chain/model left-off, reducing the burn-in effort (these may be specified as optional inputs to `fit_two_layer`). In my AL examples, I found that as few as 500 burn-in iterations were sufficient. Of course, the success of such shortcuts is intimately tied to the rejection-free nature of ESS. Efficient MCMC allows me to remove the "human-in-the-loop", specifying numbers of total and burn-in iterations in advance, without the need to investigate trace plots along the way.

### 3.1.3 Synthetic Experiments

First, I validate my DGP/AL methodology on two hypothetical simulations. To benchmark against simpler alternatives out-of-sample, I evaluate root mean-squared error (RMSE, smaller is better), and additionally a proper scoring rule [39, Eq. 25] proportional to the predictive MVN log likelihood (larger is better) in the presence of noisy simulations. I also compare the ability of the model to place design points in the region of highest complexity, offering perhaps the most compelling visual of sequential design success. I compare DGP models to both one-layer stationary GP surrogates (using MCMC sampling of hyperparameters) and non-stationary treed GP models [TGP; 44] fit through the `tgp` package on CRAN [40]. TGP uses trees to partition the input space, fits separate GPs on each rectangular partition element, and averages across an MCMC posterior sampling scheme. I use the ALC selection criterion since it is already implemented in `tgp` and allows for direct comparison.

**1d.** Consider a 1d example generated piecewise:

$$f(x) = \begin{cases} 1.35\cos(12\pi x) & x \in [0, 0.33] \\ 1.35 & x \in [0.33, 0.66] \\ 1.35\cos(6\pi x) & x \in [0.66, 1]. \end{cases} \tag{3.4}$$

Figure 3.3 provides a visual. The input space is divided into three distinct, equally-sized regions. I posit that $x \in [0, 0.33]$ is of highest modeling interest because it is the most wiggly. Realizations are observed with $\mathcal{N}(0, 0.1^2)$ additive noise. One-, two-, and three-layer models were fit via AL acquiring a total $n = 35$ runs initialized via novel $n_0 = 10$ LHSs (i.e., 25 AL acquisitions) and 100-point candidate/reference set(s) using ALC, all wrapped in a MC exercise with 100 repetitions.

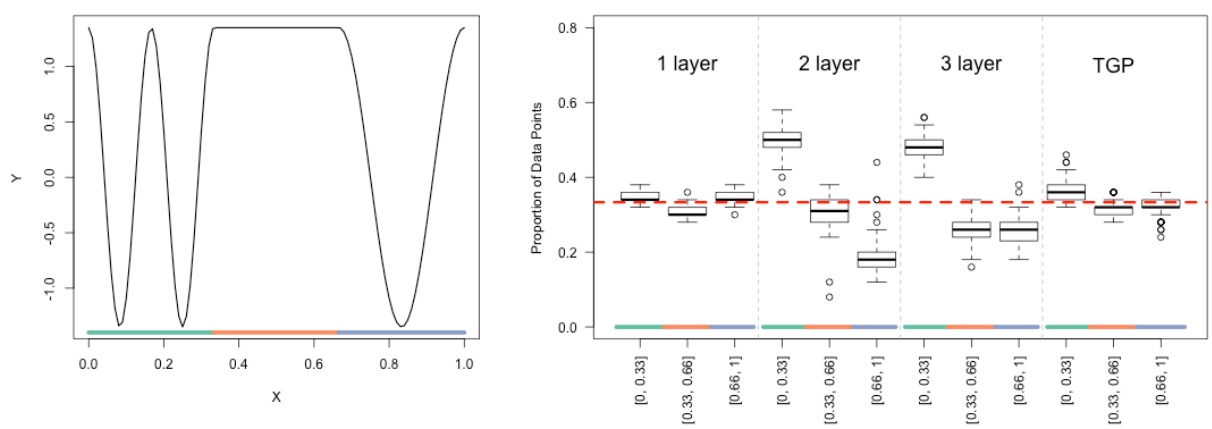


Figure 3.3: (Left) $f(x)$ from Eq. (3.4). Three regions of the input space are highlighted by the green, orange, and blue lines on the $x$-axis. (Right) Proportion of sampled points falling in each region. Boxplots represent the spread of 100 repetitions. A red dotted line marks the actual proportion of the input space occupied by each region.

Figure 3.4 shows RMSE and score calculated after each iteration for the four comparators. The two-layer DGP dramatically outperforms the one-layer GP for small data sizes ($n = 10, \ldots, 25$). Once $n$ is large enough, the one-layer model catches up. This ordinary GP

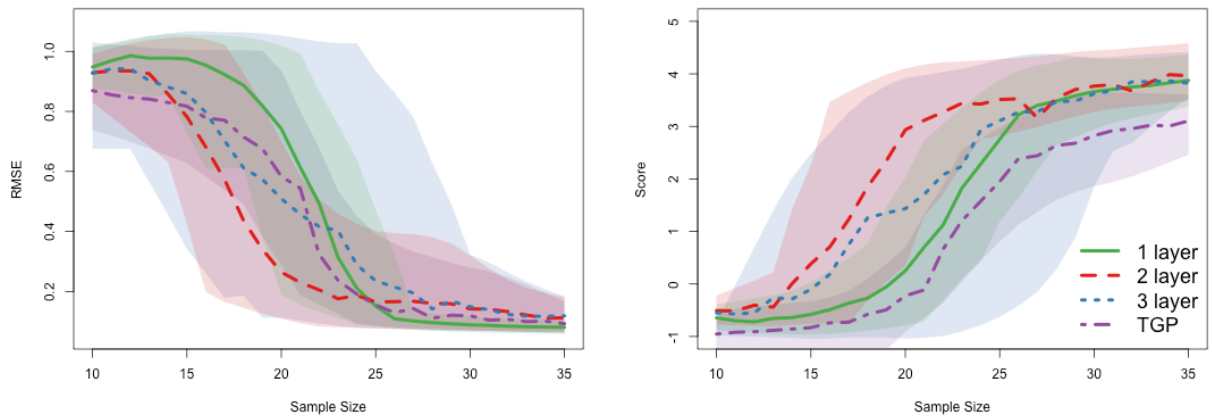


Figure 3.4: RMSE (left) and score (right) for $f(x)$ from Eq. (3.4). Solid lines represent the average over 100 repetitions. Shaded regions represent 90% quantiles.

eventually edges out on RMSE, but the two-layer DGP wins on score. The latter's wider RMSE quantiles at $n = 35$ are caused by models that over-fit the noisy observations. The three-layer model performs well on average, but is marked by very wide quantiles due to even more prominent over-fitting. Both DGPs do a better job of placing runs in the region of highest interest, $x \in [0, 0.33]$; see Figure 3.3, right. In the two-layer case, notice the right-hand region $x \in [0.66, 1]$ has the fewest acquisitions of all. This happens, I believe, because of the shape of the latent $W$ layer. Refer again to the left panel of Figure 2.3, where the right-hand region bends toward the left-hand one, "borrowing strength" from the more substantial corpus of information there. Despite a strictly distance-based kernel structure, such latent $W$ may impart a periodic effect on dynamics. The non-stationarity of the TGP model allows some departure from space-filling designs, but performs poorly on score; likely a by-product of model uncertainty in the spatial location of tree partitions.

**2d.** Next consider a 2d example originally from Gramacy and Lee [45]:

$$f(x_1, x_2) = 10x_1 \exp\left(-x_1^2 - x_2^2\right) \quad \text{for} \quad x_1, x_2 \in [-2, 4], \tag{3.5}$$

observed with $\mathcal{N}(0, 0.1^2)$ noise. Although there are no abrupt shifts, this function is characterized by two distinct regimes. Figure 3.5 marks these with color along the $x$-axes;

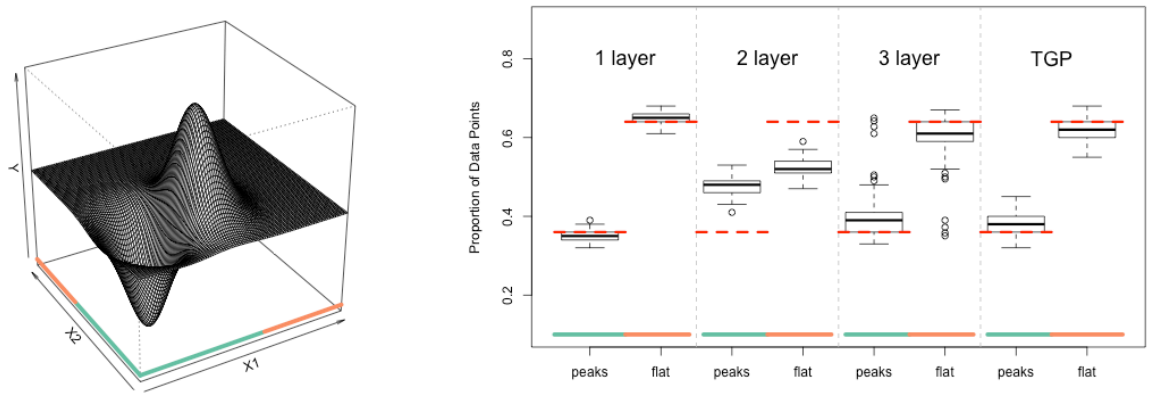


Figure 3.5: (Left) $f(x_1, x_2)$ from Eq. (3.5). Two input regions are marked by the green and orange lines along the $x$-axes. (Right) Proportion of AL acquisitions falling in each region. Boxplots indicate spread over 100 repetitions. Red dotted lines mark the actual proportions occupied by each region.

naturally the "peaky", bottom-left region is of more interest than the much larger flat one. AL is initialized with random LHSs of size $n_0 = 10$, followed by ALC acquisitions (200 candidates/references) up to $n = 80$. Figure 3.6 shows RMSE and score calculated after each

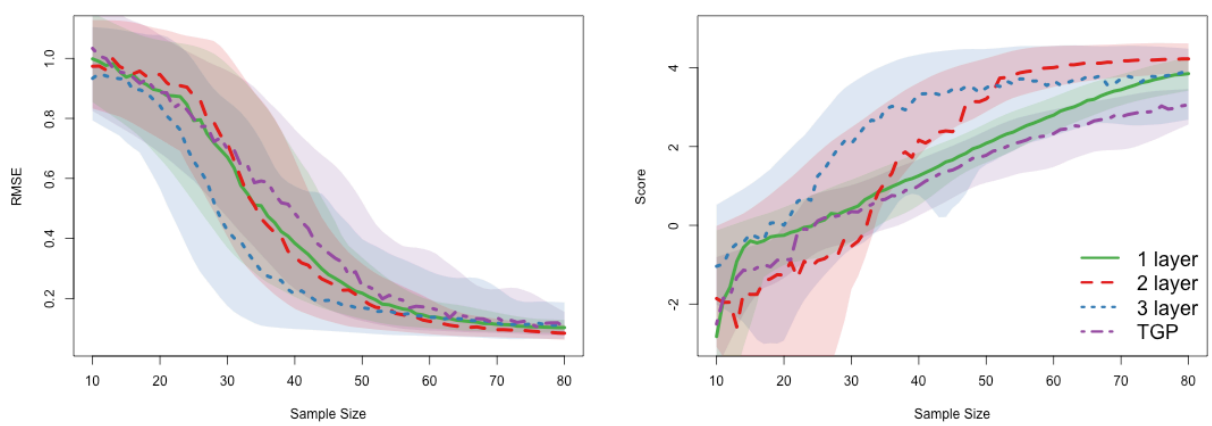


Figure 3.6: RMSE (left) and score (right) for $f(x_1, x_2)$ from Eq. (3.5). Solid lines represent the average over 100 repetitions. Shaded regions represent 90% quantiles.

iteration for one-, two-, and three-layer models as well as TGP. The two-layer DGP per-

forms similarly to the one-layer GP in terms of RMSE, but the former achieves consistently higher scores after a modest number of acquisitions ($n > 35$). The three-layer DGP beats the others in both metrics on average but has the widest quantiles. As in the 1d case, both deep models do a better job of placing design points in the region of interest; see Figure 3.5. TGP performs similarly to the one-layer GP.

### 3.1.4 Langley Glide-Back Booster Computer Experiment

The Langley Glide-Back Booster (LGBB) computer model was designed by NASA to assess the movement of a rocket booster gliding back to Earth for re-use after launching a payload into orbit. A thorough review is provided in Pamadi et al. [97]. The LGBB simulator has three inputs – speed upon entry (mach), angle of attack (alpha), and side-slip angle (beta) – and produces six deterministic response values (lift, drag, pitch, side, yaw, and roll). This model is uniquely non-stationary because the sound barrier at mach 1 imparts regime changes on aeronautic dynamics.

Gramacy and Lee [44] developed Bayesian treed Gaussian process (TGP) models specifically to account for these kinds of axis-aligned regime shifts. In my work here, I utilize a dense grid of TGP-surrogate evaluated LGBB response values from one of their AL experiments, in lieu of real simulations (the actual simulator is propriety). Having the "truth" lie inside the TGP-model class elevates this model to gold standard in terms of out-of-sample validation, but nevertheless I find comparison outside that class (with DGPs) to be informative. For more details on these data, and surrogate evaluations, see Gramacy [42, Chapter 2].

Here I focus on the side response; dynamics in the other outputs are similar. Figure 3.7 provides a visual of this response for a fixed beta value. The sound barrier causes a sharp ridge at mach 1. The region with mach values less than 2 is clearly the most "interesting".

After initializing with a random uniform (sub-) design of size $n_0 = 50$, I entertained ALC (500

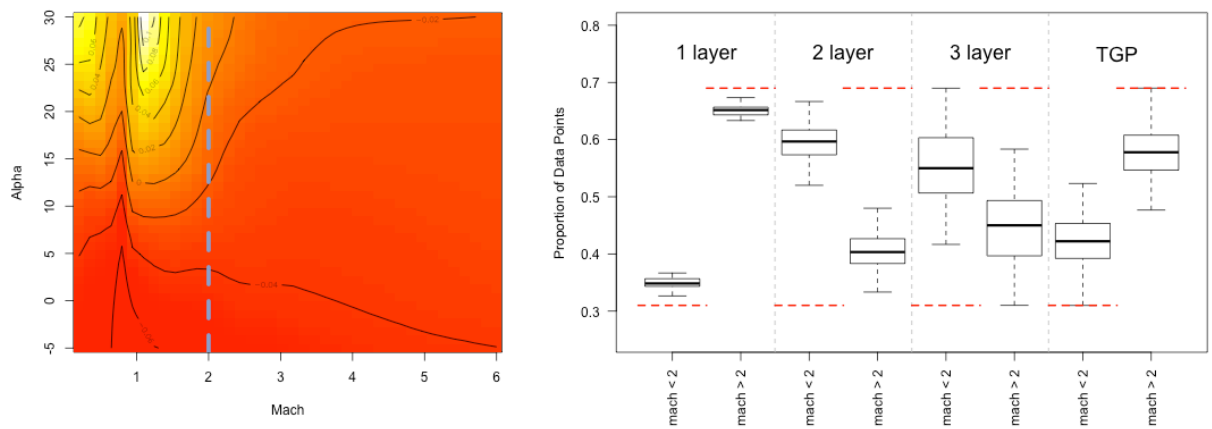


Figure 3.7: (Left) Visual of the LGBB side response for fixed beta = 4. White/high, red/low. A vertical dotted line splits the input region at mach 2. (Right) Proportion of 300 sampled points falling left/right of mach 2. Boxplots represent spread over 30 repetitions. Red dotted lines mark actual proportion of input space.

candidates/references) for acquisitions up to $n = 300$ using my usual four comparators. A nugget of $g = 10^{-8}$ was fixed to account for the deterministic nature of simulations. RMSE on a 1000-point out-of-sample testing set (newly randomized for each calculation) was evaluated after every 10th acquisition. On those occasions, I re-used the 1000-point testing set as ALC candidates/references. Joint evaluation of predictions and ALC is efficient since both rely on mapped $\mathcal{W}^{(t)}$ (2.11).

Figure 3.8 tracks out-of-sample RMSE for the four comparators over AL acquisitions, via averages over thirty MC repetitions (left), and the spread of values after the final AL acquisition (right). Both deep models outperform the one-layer GP model but are unable to match the "gold standard" of TGP. The difference in the allocation of design points among these models is pronounced. See the right panel of Figure 3.7. The one-layer model is very similar to a space-filling design. TGP deviates from space filling and puts more design points in the region of interest (mach $< 2$), but still allocates the majority of points in the flat region. DGPs even more aggressively acquire/learn in the interesting region, placing more than half

of the inputs left of mach 2.

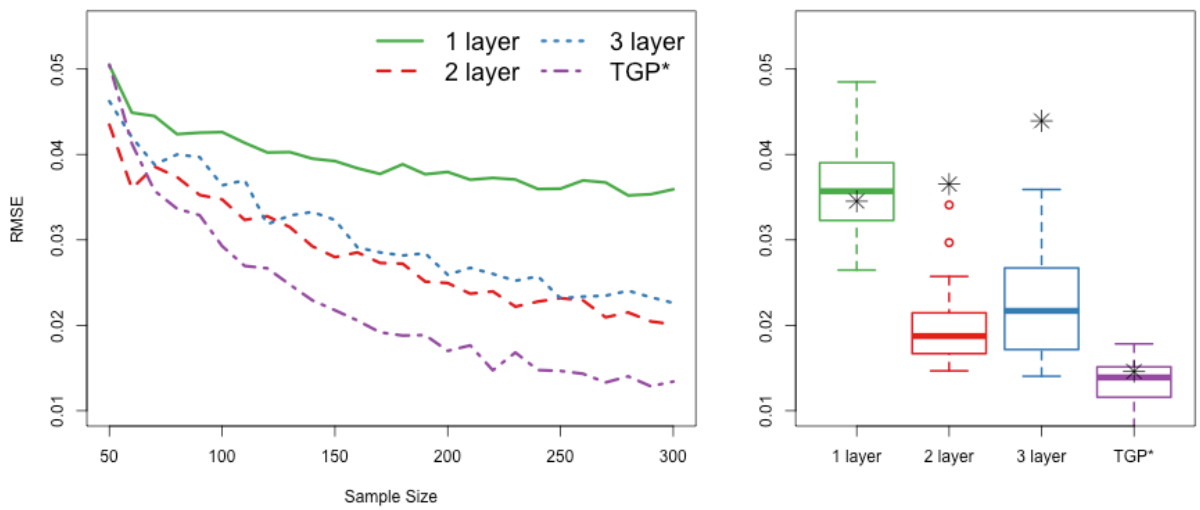


Figure 3.8: (Left) RMSE on out-of-sample 1000 point testing sets for the LGBB computer experiment, averaged over 30 repetitions. (Right) spread of final RMSE at $n = 300$ across 30 repetitions. Black stars denote the RMSE obtained from static 300-point designs. Both one-layer MLE variants performed equivalently to the one-layer MCMC fit.

To further investigate the benefit of AL with DGPs, I generated a purely random design of size $n = 300$ to train each of the four surrogates. These RMSEs are indicated as *s overlayed on the boxplots of their respective AL counterparts in the right panel of Figure 3.8. Observe that these static design results are substantially worse for the two- and three-layer fits. These DGPs demand non-uniform acquisition to predict well. One-layer results are comparable because these AL designs are essentially space-filling. The TGP results are curious. Apparently, predictive prowess here comes from "serendipitous accuracy", arising from within-model simulations, more than from AL reinforcement. Finally, I fit a one-layer model using MLE hyperparameters with both separable and non-separable lengthscales using the `laGP` package [41]. These RMSEs are identical to those obtained from the one-layer fit, with stars obscuring one another in the figure. I took this as in indication that the experiment was internally well-calibrated.

### 3.1.5 Satellite Drag Computer Experiment

Researchers at Los Alamos National Laboratory developed the *Test Particle Monte Carlo* simulator to compute drag coefficients for satellites in low earth orbit. These are useful in the development of positioning and collision avoidance systems. Sun et al. [123] created an R wrapper for the simulator and made it publicly available[1]. The simulator relies on a geometric satellite specification, atmospheric composition, and 7 input variables. See Sun et al. [123] and Gramacy [42, Chapter 2] for a thorough discussion of the simulator and its inputs. I consider the GRACE satellite here; details on atmospheric composition settings are available in my public Git repository[2]. Each simulation takes about 86 seconds on a modern desktop.

Mehta et al. [86] showed that over a restricted portion of the input space, a GP surrogate trained via 1000-point LHS is able to predict drag within 1% root mean square percentage error (RMSPE). I use Mehta's work as a benchmark and show that sequentially designed DGP models can achieve predictions within 1% RMSPE with fewer data points. I begin with a 7d LHS of size $n_0 = 100$. To help my model properly estimate the noise of the stochastic simulator I ran (multiplicity two) replicates at a random selection of ten of these original locations, so my training actually comprised of $n_0 = 110$ runs. To measure out-of-sample accuracy by RMSPE, and thus benchmark against Mehta et al., I built a single 1000-element testing set via LHS.

Considering the substantial simulator expense, I did not entertain three-layer DGPs. my previous experiments suggested that three-layer models were overkill, and potentially risky. Although their average-case behavior was attractive, it was also more volatile over repetitions. However, considering the larger input dimension compared to those earlier exercises,

[1] https://bitbucket.org/gramacylab/tpm/
[2] https://bitbucket.org/gramacylab/deepgp-ex/

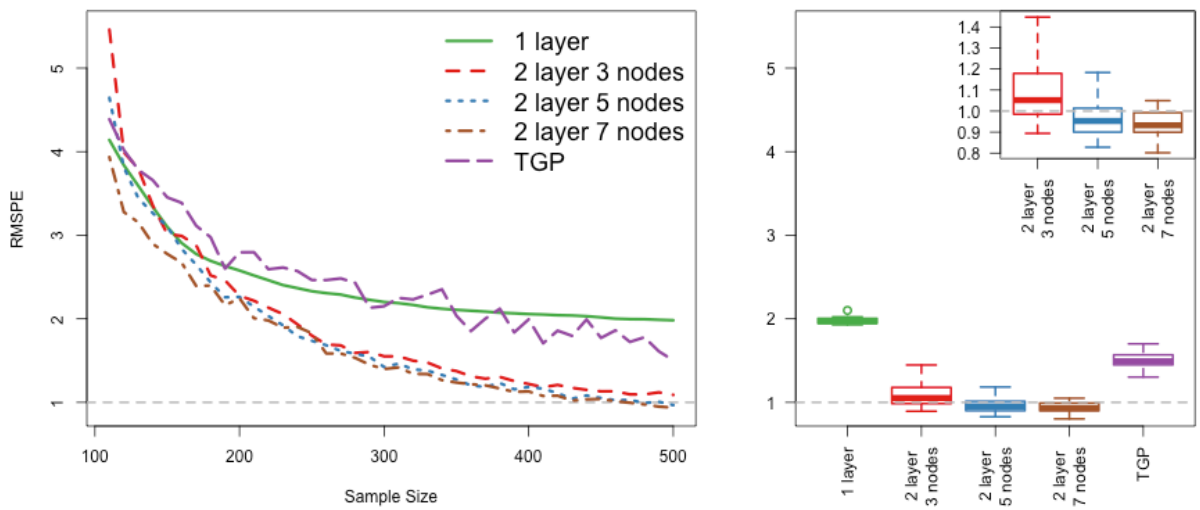


Figure 3.9: (Left) Out-of-sample average RMSPE for sequential design of the satellite drag simulator over ten repetitions. (Right) Spread of final RMSPE at $n = 500$ across 10 repetitions. Dashed lines highlight the 1% goal. The top-right sub-panel zooms in on the best comparators.

I did variations in latent dimension, with $p \in \{3, 5, 7\}$. In all three cases, training data inputs were sequentially selected using ALC (1000 candiates/references) up to $n = 500$, and RMSPE on the testing set was evaluated after every 10th acquisition. Again due to simulator expense, I limited the MC exercise to ten repetitions (only five for TGP). Results are summarized in Figure 3.9.

Observe that all but the one-layer (ordinary GP) and TGP comparators were able to achieve the 1% benchmark, with high probability, with fewer than 500 runs of the simulator. Interestingly, it is possible to accomplish this feat with a latent dimension of $p = 3 \ll d = 7$. Although results are improved with more latent dimensions, up to $p = d = 7$ which is the software default, this comes at greater computational expense ($p = 3$ is two times faster on an eight core machine).

## 3.2 Bayesian Optimization

Optimization is a common surrogate modeling objective; the goal is to find the input configuration that yields the lowest response value (alternatively, to find the highest response simply assign $f(x) \to -f(x)$). More concretely, one seeks to solve the global optimization problem

$$x^\star = \underset{x \in B}{\operatorname{argmin}} \; f(x) \quad \text{for bounding box } \; B \subset \mathbb{R}^d, \tag{3.6}$$

ideally with as few evaluations of the blackbox computer experiment as possible. Greedy sequential design that leverages surrogate information to seek $x^\star$ (3.6) is commmonly referred to as "Bayesian optimization" (BO). Rather than targeting posterior variance as in Section 3.1, acquisitions seek to balance exploitation of known "low" areas and exploration of high variance regions. Popular BO acquisition criteria include expected improvement [EI; 60], Thompson sampling [126], and the upper confidence bound [1]. I prefer the EI criterion because it has a deterministic, closed-form analytic expression, which lends well to evaluation with MCMC-based DGPs.

Consider the simple one-dimensional example provided in Figure 3.10. Red circles indicate training data ($n = 6$). Predicted mean and 95% confidence intervals from a one-layer GP (using the `fit_one_layer` functionality in the `deepgp` package) are shown by the solid and dashed black lines. Each grey line represents a random draw from the posterior distribution $Y(\mathcal{X}) \mid D_n$ following Eq. (1.4). The lowest observation so far is the right-most red point; this is termed the "best observed value" and is denoted as $f_{\min}$ (assuming $f$ is deterministic – I will discuss adaptation to noisy data in Section 3.2.2). The next acquisition $x^\star$ would be an improvement if $f(x^\star) < f_{\min}$. There are two promising domains where a lower value may be observed: in the middle of the space near $x = 0.5$ and at the upper boundary of the space near $x = 1$. These two realms have low predicted mean and high predictive variance.

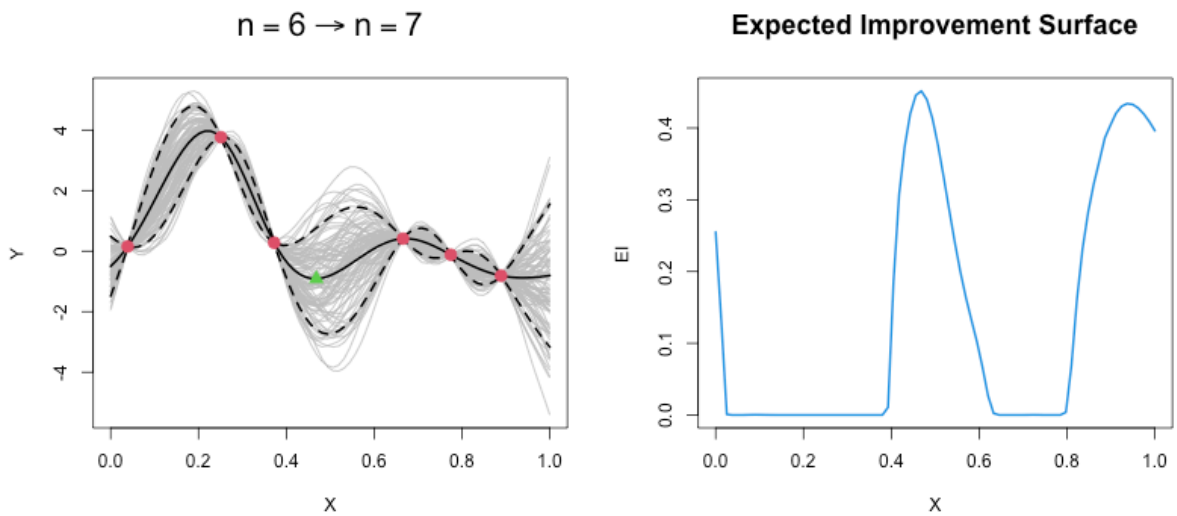


Figure 3.10: (Left) One-layer GP predicted mean (solid black), 95% confidence interval (dashed black), and posterior draws (grey). Red circles indicate training data; green triangle indicates EI acquisition. (Right) The EI criterion evaluated along a dense grid.

Let $\mu_Y(x)$ and $\sigma_Y(x)$ denote posterior predictive mean and standard deviation at location $x$ (Eq. (1.4) with singleton $\mathcal{X} = x$). The EI criterion is defined as

$$\mathrm{EI}(x) = (f_{\min} - \mu_Y(x))\, \Phi\left(\frac{f_{\min} - \mu_Y(x)}{\sigma_Y(x)}\right) + \sigma_Y(x)\phi\left(\frac{f_{\min} - \mu_Y(x)}{\sigma_Y(x)}\right) \tag{3.7}$$

where $\Phi$ is the standard Gaussian CDF and $\phi$ is the standard Gaussian PDF. EI is high when $\mu_Y(x) < f_{\min}$ (i.e., exploitation) and/or $\sigma_Y(x)$ is high (i.e., exploration). The right panel of Figure 3.10 provides a visual of the EI surface for the GP surrogate fit of the left panel. The green triangle in the left panel indicates the location that maximizes EI. BO would progress by evaluating the blackbox simulator at this location and repeating the process of updating the surrogate and optimizing EI, following the procedure of Figure 3.1.

The effectiveness of EI-acquisitions is intricately tied to the fidelity of the surrogate posterior predictions. If the true surface is non-stationary, a DGP surrogate may provide better predictions and more effective BO than a one-layer stationary GP. Hebbal et al. [53] utilized

DGPs for BO, but with variational inference and inducing point approximations (tools which oversimplify UQ and are lower fidelity than the alternatives I propose, see Chapter 4 for further discussion and demonstration). Posterior variance estimates are just as prominent in the EI criterion (3.7) as posterior mean estimates. Even a flexible DGP will not be a useful BO tool without both high accuracy and full UQ. My fully-Bayesian MCMC-based DGP (Chapter 2) thus has potential to excel in non-stationary BO settings.

Up to this point, I have glossed over the most challenging part of the BO procedure: finding the maximum EI location. The EI surface is generally multi-modal with many local optima. For a visual, return to the EI surface in the right panel of Figure 3.10. There are three local maxima: near $x = 0$, $x = 0.5$, and $x = 0.95$. A numerical optimizer is likely to get stuck in inferior modes depending on where it is initialized. To circumvent this, multi-start optimization is required: run many instances of a numerical optimizer initialized at different locations and select the highest of all local optima identified. Yet multi-start optimization is very computationally cumbersome. Each optimization requires a large (and possibly unconstrained) number of simulatior evaluations, many of which must proceed sequentially, without scope for parallelization. Such schemes are especially infeasible for MCMC-based surrogates.

The alternative to numerical optimization is candidate optimization: evaluate EI at a smattering of input locations (which may be done simultaneously, leveraging parallel computing) and acquire the location that yields the highest EI. The fidelity of candidate optimization relies heavily on the placement of candidates. Candidates that are allocated randomly or based on space-filling LHS (as I used in Section 3.1) are likely to miss the steep peaks of the EI surface, especially in higher dimensions.

### 3.2.1 Triangulation Candidates

The work in this section stems from joint work with Bobby Gramacy and Nathan Wycoff [46]. I present a new scheme for allocating candidates for BO with large MCMC-based DGPs when numerical optimization of the sequential design criterion is not feasible. Start with a *Delauney triangulation* of the existing training data locations: a set of line segments between points that divy up the space so no lines cross. Figure 3.11 provides a visual of this in two dimensions. Open circles indicate observed training data locations, and black solid lines form the Delauney triangulation. In two dimensions, this triangulation forms triangles; in higher dimension it forms tetrahedra although I will still refer to them as triangles. Each triangle is defined by a $(d+1) \times d$ matrix. Denote each triangle by $T_j$ for $j = 1, \ldots, n_T$. The number of triangles, $n_T$, is affected by the size of the training data, $n$, and the dimension of the input space, $d$.

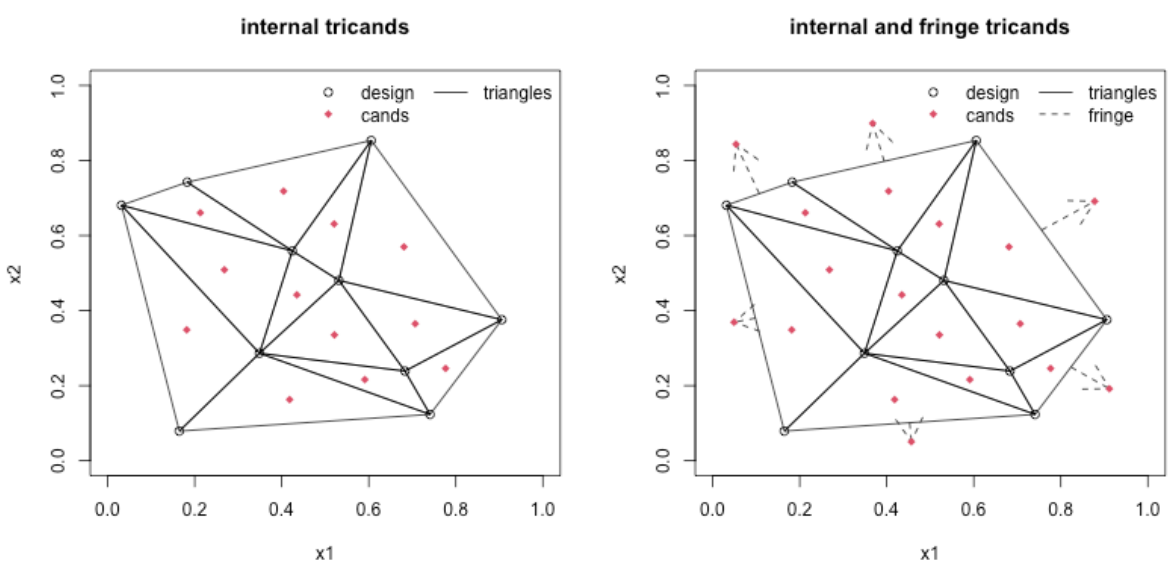


Figure 3.11: Internal and fringe ($\alpha = 0.5$) triangulation candidates.

I allocate candidates in two realms: internal and fringe. For internal candidates, I propose

the *barycenter* of each triangle,

$$\tilde{x}_j = \frac{1}{d+1} \sum_{i=1}^{d+1} T_j[i,].$$

These candidates are indicated by the red diamonds in the left panel of Figure 3.10. Notice how they are spread out between existing design points. For fringe candidates, denote $F_j$ for $j = 1, \ldots, n_F$ as the $d \times d$ matrix representing a facet (outer edge of the triangulation). Together, these facets form the convex hull. Calculate the midpoint of each facet as $\bar{F}_j = \frac{1}{d} \sum_{i=1}^{d} F_j[j,]$ then take the normal vector $\vec{v}_j$ extending perpendicularly from $F_j$ at the midpoint $\bar{F}_j$. I allocate a candidate along each $\vec{v}_j$, with the exact placement determined by a tuning parameter $\alpha \in [0, 1]$ representing the proportional distance from the facet midpoint to the boundary of the space. For BO, I have found $\alpha = 0.5$ to perform well. Fringe candidates for the example of Figure 3.10 are represented by the tips of dashed arrows. I refer to the combination of internal and fringe candidates as the "tricands."

As dimension increases, the number of total candidates, $N = n_T + n_F$ will increase. For large $N$, it is helpful to constrain the computational burden by setting a maximum size of the candidate set. I accomplish this with targeted sub-sampling of the triangulation candidates. In BO settings, I subsample a maximum set of $100 \times d$ candidates, making sure that atleast 10% of the retained candidates are near the current best observed value to ensure opportunities for exploitation. See Gramacy et al. [46] for details. Overall, this tricands scheme allows for strategic allocation of points betweeen existing training design locations (and the bounding box), which is where sequential design criteria optima are most likely to be found.

I use the `geometry` package for R [50], which is coded in C under the hood, to compute Delauney triangulations. I provide R code wrappers for calculating tricands with all of the

nuances I have described thus far in a public git repository[3].

### 3.2.2 Implementation Details

With EI criterion (3.7) and candidate scheme in hand, integration with the Bayesian DGP is relatively straightforward. Start with an initial training design and conduct MCMC sampling for DGP latent variables following Section 2.2. Generate tricands candidates, $X_{\text{tri}}$ as described in the previous subsection. For each burned-in sample $t \in \mathcal{T}$, calculate posterior predictive mean and point-wise variances, $\mu_Y^{(t)}(X_{\text{tri}})$ and $\sigma_Y^{2(t)}(X_{\text{tri}}) = \text{diag}(\Sigma_Y^{(t)}(X_{\text{tri}}))$ following Eqs. (2.11–2.12) (by mapping through latent warpings) at $X_{\text{tri}}$ locations. Then slot these into the EI calculation,

$$\text{EI}^{(t)}(X_{\text{tri}}) = \left(f_{\min} - \mu_Y^{(t)}(X_{\text{tri}}))\right) \Phi\left(\frac{f_{\min} - \mu_Y^{(t)}(X_{\text{tri}})}{\sigma_Y^{(t)}(X_{\text{tri}})}\right) + \sigma_Y^{(t)}(X_{\text{tri}})\phi\left(\frac{f_{\min} - \mu_Y^{(t)}(X_{\text{tri}})}{\sigma_Y^{(t)}(X_{\text{tri}})}\right),$$

and take the expectation over $t \in \mathcal{T}$,

$$\text{EI}(X_{\text{tri}}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \text{EI}^{(t)}(X_{\text{tri}}).$$

When the simulation is deterministic, $f_{\min} = \min(Y_n)$ is a true representation of the best observed value (BOV). But if the simulation is observed with stochastic noise, then $\min(Y_n)$ may not represent the actual best location, particularly if the low value was simply caused by white noise. In this case, I implement the replacement of $f_{\min}$ with $f_{\min}^{(t)} = \min(\mu_Y^{(t)}(X_n))$. In other words, find the predicted mean at the training data locations and use the minimum mean value as the BOV, thus removing the impact of the noise term. In the case of the DGP, these predicted BOV's must also be indexed based on the MCMC sample and averaged appropriately. In the presence of noise, I remove the additive nugget from the point-wise

[3] https://bitbucket.org/gramacylab/tricands/

posterior variances (i.e. remove $g^{(t)}\mathbb{I}_{n'}$ in Eq. (2.12)).

EI calculations are offered as an optional feature of the `predict` commands in the `deepgp` package. They are triggered by the indicator `EI = TRUE`.

```
R> fit.2 <- predict(fit.2, Xtri, EI = TRUE)
```

Optional parallelization via `foreach` [88] constructs is triggered by additionally setting the `cores` argument to an integer greater than one.

### 3.2.3 Synthetic Experiment

As an empirical comparison, I offer a synthetic example involving the Levy function, which is again found in the Virtual Library of Simulation Experiments [124] and is defined in arbitrary dimension,

$$f(x) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2\left[1 + \sin^2(2\pi w_d)\right]$$
$$\text{where} \quad w_i = 1 + \frac{x_i - 1}{4} \quad \text{for all} \quad i = 1, \ldots, d.$$

Here I use $d = 5$ and a random starting design of size $n_0 = 12$. I entertain two non-stationary surrogates: my Bayesian DGP and the treed Gaussian process [TGP; 44] implemented in the `tgp` package [40]. I entertain two candidate based allocations: Latin Hypercube samples (lhs) and my tricands (tri). The maximum candidate set size for both is fixed at $N = d \times 100 = 500$.

Results for EI-based acquisitions up to $n = 75$ are provided in Figure 3.12. The left panel reports the median BOV over 100 Monte Carlo repetitions (with re-randomized initial designs). The center panel shows a slice of the performance midway through the design at

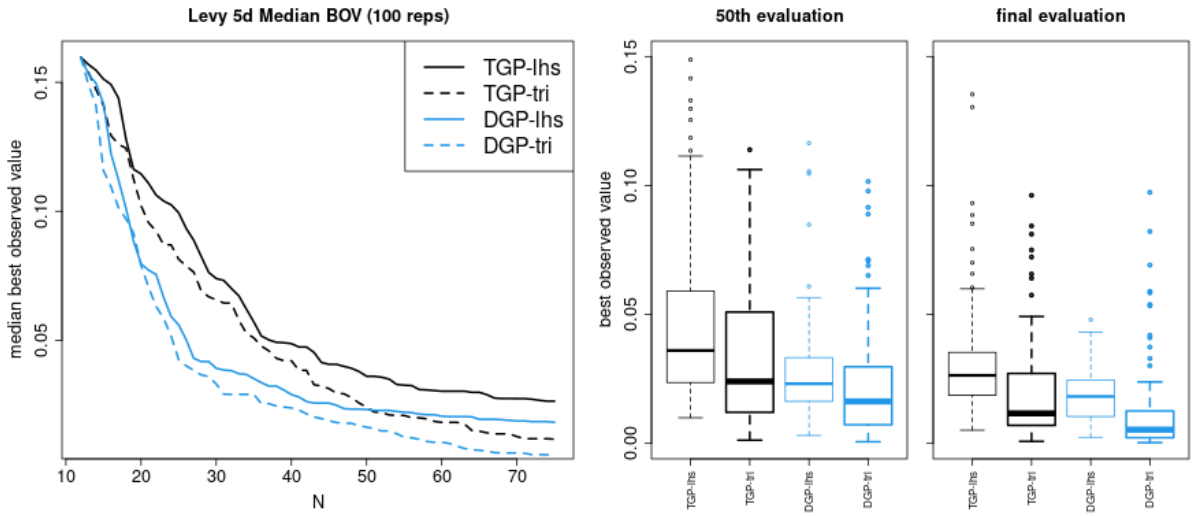


Figure 3.12: Bayesian optimization of the Levy function in 5d.

$n = 50$. The right panel shows the performance at the end of the sequential design. The DGP outperforms the TGP competitor in this example, but the key takeaway is that both surrogates perform better with tricands candidates than LHS candidates (in the left panel, the dotted lines are uniformly dominated by the solid lines).

## 3.3 Contour Location

Another common surrogate modeling objective is "contour location" (CL) in which the goal is to identify an entire level set in the response surface. Contour location goes hand-in-hand with reliability analysis; the specified level set denotes the boundary between a pass and fail ("pass" and "fail" definitions are domain specific, they may refer to safe v. unsafe conditions or efficient v. inefficient use). In practice, it is crucial to know the precise locations of failure regions so they can be avoided. These situations are especially prevalent in aerospace enginnering applications in which response values are linked to either safety or performance characteristics of aircraft.

Contour location is very similar to Bayesian optimization; both seek to sequentially select inputs that balance exploitation and exploration. The key difference lies in the size of the target: in BO the target is a single location, in CL the target is an entire slice of the surface. As a concrete demonstration, consider the surface displayed in Figure 3.13. The formal definiton of the displayed function is provided in Eq. (3.10), but I will simply refer to it as the "plateau" function. It is marked by two flat regions with a sloping drop between them. I have defined a failure boundary at $f(x) = 0$, indicated by the solid blue line in the left panel. The sequential design objective is to pinpoint the location of this entire line in as few evaluations of the blackbox function as possible.

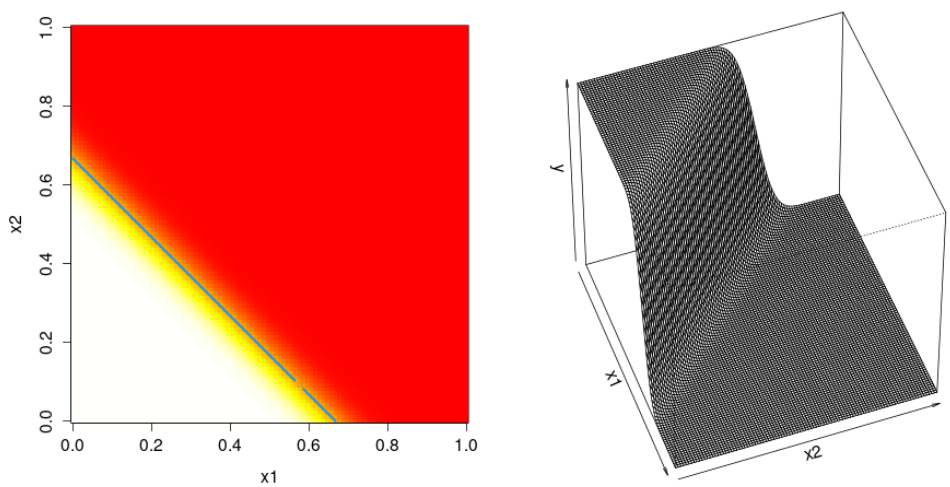


Figure 3.13: Heat map and surface plot of the 2-dimensional plateau function (3.10).

Existing work on surrogate-informed contour location utilizes GP surrogates almost exclusively (I have not seen DGPs deployed in this context yet). The most popular acquisition criterion for contour location is entropy [17, 82, 93]. Let $g$ indicate the contour level threshold such that $f(x) > g$ indicates a failure. Denote $p_x = \mathbb{P}(f(x) > g)$ as the probability of observing a failure at location $x$. The entropy criteria,

$$\text{Ent}(x) = -p_x \log(p_x) - (1 - p_x) \log(1 - p_x), \tag{3.8}$$

will be high in regions of pass/fail uncertainty (i.e. $p_x \approx 0.5$). Under a GP surrogate, failure probabilities simplify to a Gaussian CDF computation,

$$p_x = \mathbb{P}\left(f(x) > g\right) = 1 - \Phi\left(\frac{g - \mu_Y(x)}{\sigma_Y(x)}\right), \tag{3.9}$$

where $\mu_Y(x)$ and $\sigma_Y(x)$ are the predicted posterior mean and standard deviation respectively. In a DGP, the posterior predictions are conditionally Gaussian instead of margianally Gaussian, but I find the form of Eq. (3.9) sufficient nonetheless.

To provide a concrete visual, I fit a two-layer DGP to random training data of size $n_0 = 15$ for the plateau function. [I'm not providing a comparison to a stationary GP fit at this point, but hopefully I've convinced you by now that a one-layer GP would struggle to handle the steep transition between the two flat regions.] The left panel of Figure 3.14 shows a heatmap of the predicted mean surface, with training data locations indicated by open circles. The estimated failure contour is displayed by the black dashed line; the true failure contour is overlayed in blue for reference. The center panel displays a heat map of the predicted standard deviation. Notice that the predictive uncertainty is high in areas near the transition and far from training data [If I had fit a stationary GP here, this plot would look just like the left panel of Figure 3.2 with uncertainty solely based on distance to training data.] The right panel displays a heat map of the entropy surface. It is very peaky, with a ridge of maxima highlighted by the blue circles. For now, ignore the green diamond, I will discuss it in Section 3.3.1.

Two of the high entropy points (blue circles) are nestled right next to existing training data locations. Even though uncertainty is low here, entropy is still high. And this is the region where the surrogate actually predicts the contour most accurately! An acquisition in this region would not be advantageous. Simply maximizing entropy will not yield the

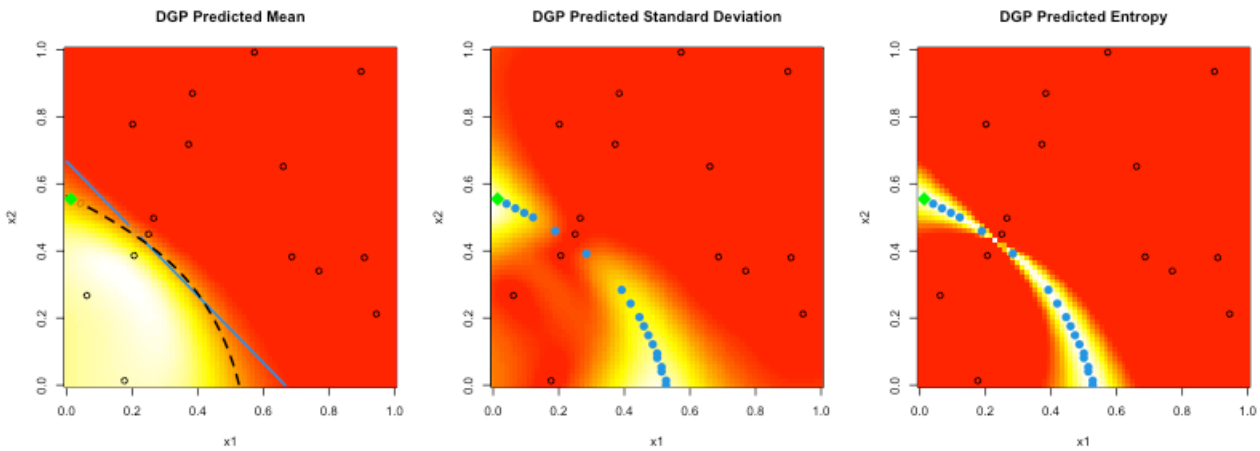


Figure 3.14: Two-layer DGP predicted mean (left), standard deviation (center), and corresponding entropy (right). Open circles indicate training data. Blue circles indicate locations with highest entropy. Green diamond indicates Pareto front acquisition (Section 3.3.1). White/high, red/low.

best acquisition. Cole et al. [17] noticed this inconvenient behavior of entropy and posited a solution that seeks local optima in the entropy surface, as an alternative to the global optima, through a hybridized multi-start optimization scheme. While this is a step in the right direction, this strategy still relies on entropy as the sole acquisition criteria and results in acquisitions that cluster together. Nevertheless, Cole et al. demonstrated superior performance over a plethora of competing methodologies, making their method my primary competitor.

### 3.3.1 Pareto Front of Entropy and Uncertainty

In this section I propose a novel sequential design scheme for DGP-informed contour location. The motiviation behind this contribution is two-fold. First, I need an acquisition scheme that relies only on prediction at candidate locations and not on burdensome numerical optimization (which is incompatible with the Bayesian DGP). Second, I need an acquisition criterion that addresses the pitfalls of entropy (it is overly peaky and does not thoroughly

promote exploration).

To motivate my proposed methodology, let us return to the example of Figure 3.14. An ideal acquisition would acquire a new input on the outer edges of the predicted contour where the contour prediction is not accurate *and* surrogate uncertainty is high. In summary, I want to target both high entropy and high uncertainty, not one or the other. The left panel of Figure 3.15 combines the visuals from the center and right panels of Figure 3.14 by plotting entropy vs. predictive standard deviation, evaluated over a dense grid of locations. An ideal acquisition would be in the upper right corner of this plot. This "upper-right edge" is the "Pareto" front (red triangles). Acquisitions along that frontier target both high uncertainty and high entropy.

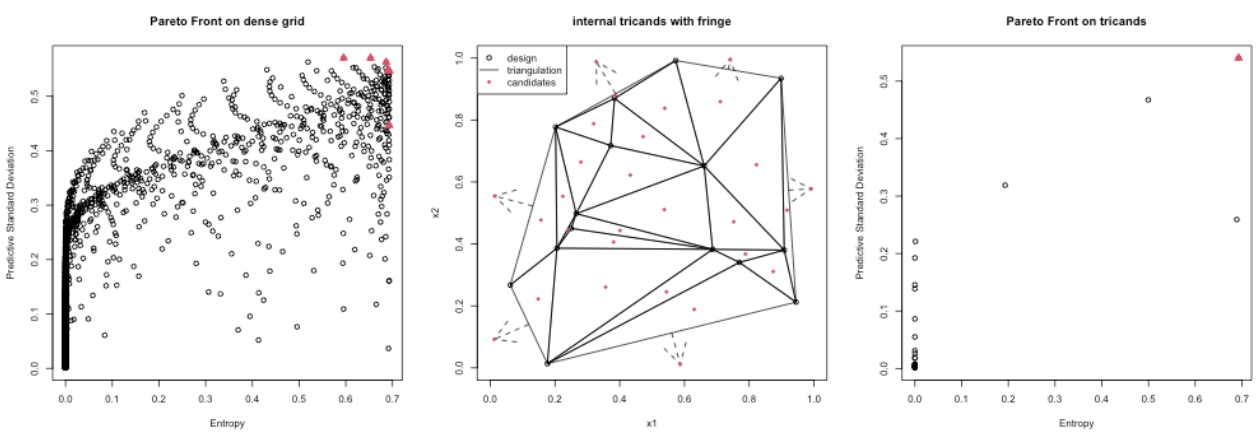


Figure 3.15: (Left) Entropy vs. predictive standard deviation for the DGP of Figure 3.14. Red triangles indicate the "Pareto" front. (Center) Triangulation candidates ($\alpha = 0.9$) for the existing design. (Right) Same as left, but evaluated at tricands instead of a dense grid.

My candidate-based contour location scheme boils down to three steps: propose a selection of candidate locations, use the surrogate to evaluate posterior standard deviation and predictive entropy for these candidates, and select an acquisition on the Pareto front of uncertainty and entropy. I have yet to discuss the first of these steps though. In higher dimension it is not possible to propose a dense grid of candidates. Even in lower dimensions, there is a computational limit to the number of candidates that can be quickly evaluated through the

surrogate.

Thus, I again need a strategic method for allocating candidates... re-enter tricands! Triangulation candidates (Section 3.2.1) strategically allocate candidates between existing training data, where predictive uncertainty is likely to be high, making them a natural fit for my Pareto front acquisitions. To tailor tricands for CL (instead of BO), I offer two adaptations. First, I utilize $\alpha = 0.9$ for fringe candidates to push candidates closer to the boundary. I find this is advantageous since failure regions are more likely to be near boundaries (as opposed to optimum locations which are more commonly in the interior of the space). Second, I adapt the targeted sub-sampling approach to guarantee at least 10% of sub-sampled tricands are near high response locations (instead of the "best observed value" I used previously).

The center panel of Figure 3.15 displays tricands (red diamonds and points of arrows) for the training data of the plateau function. The right panel re-creates the left panel, but with evaluations done only on the tricands locations, with the Pareto front acquisition marked by the red triangle. The right panel is much sparser, but it still manages to place an acquisition in a strategic location. Returning to Figure 3.14, this tricands-Pareto-front acquisition is indicated by the green diamond. It is in an ideal area of high entropy and high uncertainty.

### 3.3.2 Implementation Details

I utilize the `deepgp` package for DGP surrogate training and predictions. The calculation of entropy is straightforward, given posterior predictive moments (3.8–3.9). An updated version of tricands that accounts for targeted sub-sampling near a contour is provided in the same git repository.[4] I utilize the `rPref` `R` package on CRAN [105] for identification of the Pareto front.

[4] https://bitbucket.org/gramacylab/tricands/

```
R> XPareto <- psel(data.frame(ent, sigma), pref = high(ent) * high(sigma))
```

When the Pareto front includes more than one candidate location (as in the multiple red triangles of the left panel of Figure 3.15), I sample an acquisition from these uniformly at random. Reproducible code that runs an entire sequential design is provided in my other git repository (the same one I've been plugging all along).[5]

### 3.3.3 Synthetic Experiments

I entertain the following competitors:

- DGP ESS Pareto: my Bayesian ESS-based DGP using tricands and the Pareto front acquisition defined in Section 3.3.1.

- GP MCMC Pareto: a stationary one-layer GP with MCMC-sampled separable lengthscales utilizing tricands and the Pareto front acquisition.

- GP MLE Pareto: a stationary one-layer GP utilizing maximum likelihood estimated separable lengthscales and the multi-start pseudo optimization of entropy from Cole et al. [17].

All surrogates utilize the Matèrn $\nu = 5/2$ kernel. The first two competitors differ only in the use of a DGP surrogate, while the third competitor represents the "state-of-the-art" at the time of publication. All empirical studies in this section are observed without noise.

[5] https://bitbucket.org/gramacylab/deepgp-ex/

**Plateau Function.** Consider the function

$$f(x) = 2 * \Phi\left(\sqrt{2}\left(-4 - 3\sum_{i=1}^{d} x_i\right)\right) - 1 \quad \text{for } x \in [-2,2]^d \ \left(\text{but scaled to } x \in [0,1]^d\right), \tag{3.10}$$

which I have adapted from Izzaturrahman et al. [57] to be defined in arbitrary dimension. I define the failure contour at $g = 0$, such that $f(x) > 0$ indicates a failure. Starting with $d = 2$ and an initial LHS design of size $n_0 = 5$, I conduct sequential designs targeting this contour up to an ending design size of $n = 30$ (25 acquisitions). The median sensitivity or "true failure rate" (1.0 indicates detection of all failures) across 50 Monte Carlo (MC) repetitions with re-randomized starting designs is displayed in the left panel of Figure 3.16. The distribution of sensitivities across MC repetitions at the end of the sequential design are displayed in the left side of the right panel. As an additional benchmark, I fit each surrogate on static LHS designs of equivalent size ($n = 30$) and reported the sensitivities in the same panel.

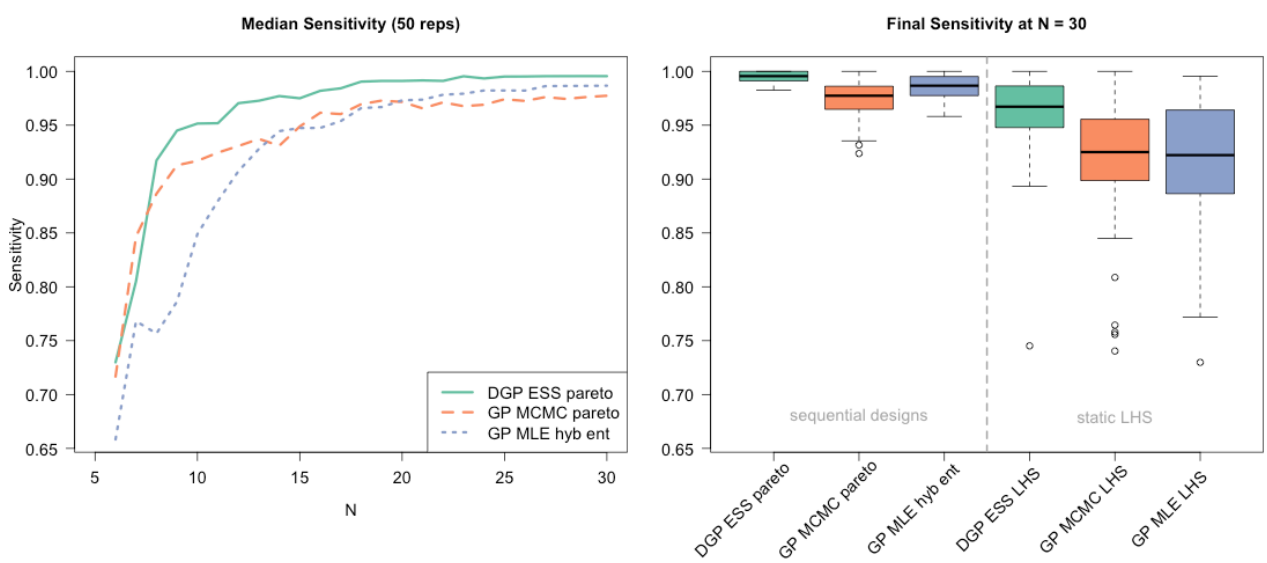


Figure 3.16: Sensitivity (higher is better) for the two-dimensional plateau function.

Notably, although the DGP surrogate starts at a similar place as the MCMC-based GP sur-

rogate, it quickly jumps to the top performance after a couple of acquisitions and maintains this superiority for the extent of the design. All methods benefited from the contour locating sequential designs when compared to their static LHS counterparts.

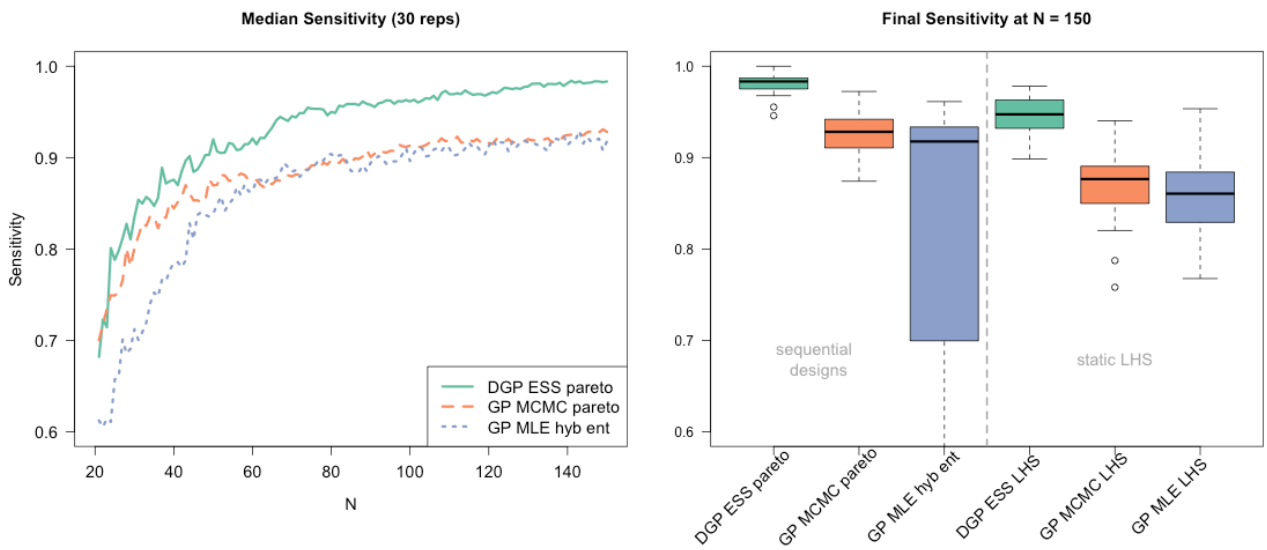


Figure 3.17: Sensitivity for the five-dimensional plateau function.

To make things a little tricker, I bumped the dimension up to $d = 5$ and repeated this exercise. Higher dimension requires larger data sizes, so I started with $n_0 = 20$ and extended the designs up to $n = 150$. Results are shown in Figure 3.17. Performance trends are similar to those of Figure 3.16, although it appears the DGP has an even bigger edge in this higher dimensional setting. Also noteworthy, the entropy-based acquisitions of the "GP MLE hybrid" model occasionally led the GP astray, and offered poorer performance than their static design counterparts. I supsect this is due to clustering of points and a lack of exploration.

**Cross-in-Tray Function.** Next, consider the "cross-in-tray" function

$$f(x_1, x_2) = -0.001 \left( \left| \sin(x_1)\sin(x_2)\exp\left( \left| 100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi} \right| \right) \right| + 1 \right)^{0.1}$$

found on the pages of the VLSE [124]. I use the domain $x \in [-2, 2]^2$ (similarly scaled down to $[0.1]^2$). A visual of the surface is provided in the left panel of Figure 3.18 with a contour defined at $g = 2$. I follow the same sequential design procedures, but increase training data sizes to $n_0 = 50$ and a final design size of $n = 300$ to account for the increased complexity in the surface and the contour. Results over 20 MC repetitions are displayed in Figure 3.19. Again, all methods benefit from the sequential design, but the DGP has a clear advantage due to its non-stationary flexibility.

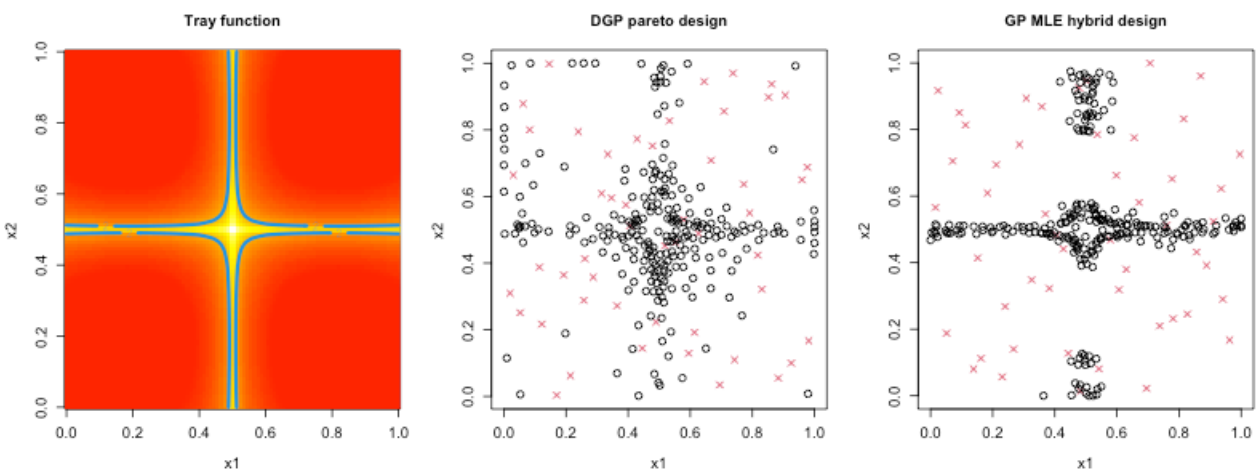


Figure 3.18: (Left) Heat map of the two-dimensional cross-in-tray function with contour at $g = 2$. (Center) Sequential design from two-layer DGP with Pareto front acquisitions. (Right) Sequential design from stationary GP with hybrid-entropy acquisitions. Red "x" indicate starting LHS; black circles indicate acquired points.

In addition, the center and right panels of Figure 3.18 display the resulting design at the end of a single MC exercise for the "DGP ESS Pareto" scheme and the "GP MLE hybrid" scheme. Red "x" indicate the starting LHS design, and black circles indicate the acquired points. As expected, the acquisitions that rely solely on entropy tend to cluster (right panel). The Pareto front acquisitions do a better job of exploring the contour. The real "secret sauce" though is the combination of the Pareto front acquisition scheme with the right surrogate model (the flexible two-layer DGP in this case).

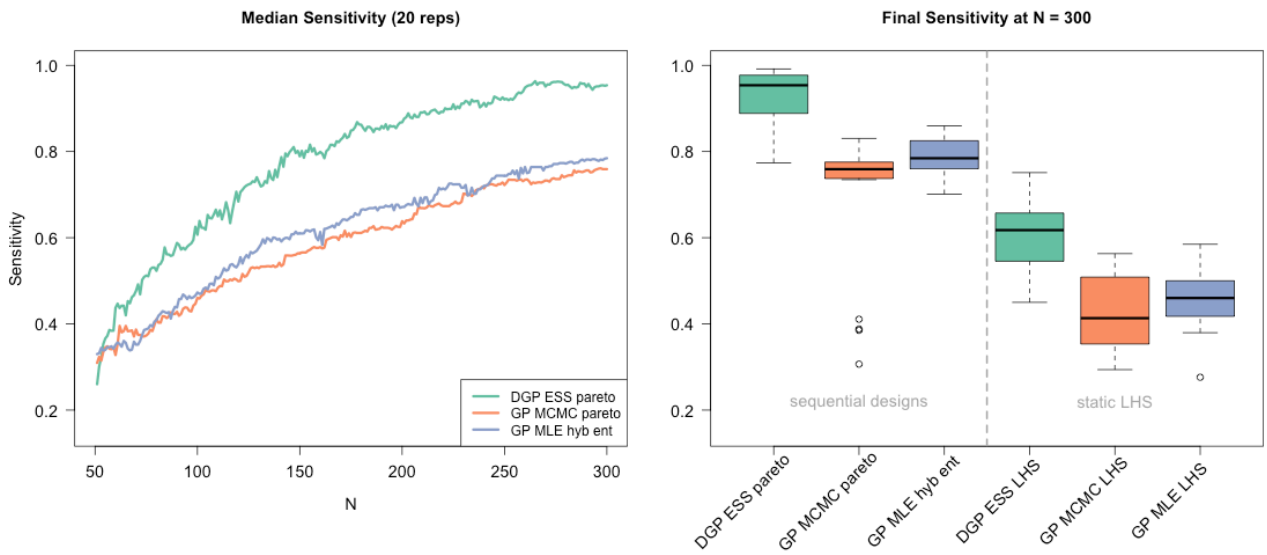


Figure 3.19: Sensitivity for the two-dimensional cross-in-tray function.

### 3.3.4 SU2 Airfoil Computer Experiment

The SU2 suite provides an array of aeronautic computer experiments [28]. I am particularly motivated by a 7-dimensional computational fluid dynamics simualtion of incompressible flow around an airfoil. Four of the seven inputs are wing shape parameters; they define the airfoil mesh. The other three inputs are angle of attack, Reynolds number (which controls fluid dynamics characteristics), and mach number (speed). The response variable is the "lift to drag ratio" (LD). High LD values correspond to high efficiencies; an aircracft will not require as much fuel in these settings. Low LD values are inefficient. Ideally, wing design and flight characteristic (like angle and speed) will be calibrated to avoid inefficient settings.

DGPs are relatively new to this area, but expert knowledge suggests that there is some non-stationarity in the response surfaces of these simulations. To test this, I ran the simulator at a large LHS of size $n = 5,000$. It takes about 5 minutes to get a single observation on an 8-core hyper-threaded Linux machine, so evaluating this entire design required weeks of compute time. I took random subsets of size $n = 500$, fit both two-layer DGP and stationary

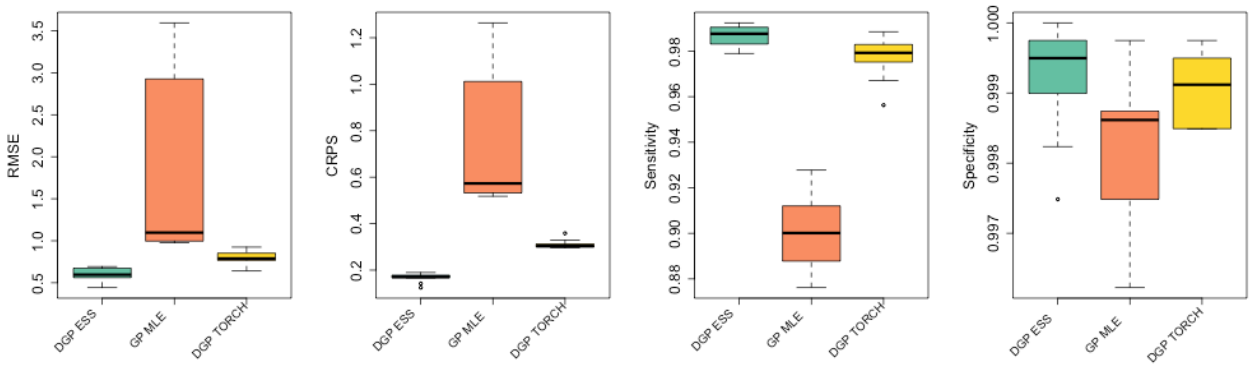


Figure 3.20: Root mean squared error, continuous rank probability score [39], sensitivity, and specificity for surrogate fits to static LHS designs of size $n = 500$. Failures defined at LD $< 3$. Lower RMSE/CRPS and higher sensitivity/specificity are better. Boxplots show 10 MC repetitions.

one-layer GP surrogates, and evaluated their performance on the remaining 4,500 points. In addition to my Bayesian ESS DGP ("DGP ESS"), I used the `gpytorch` suite in `Python` [36] to fit a two-layer DGP ("DGP TORCH") that utilizes variational inference and inducing point approximations. The `gpytorch` implementation required careful, tedious tuning of the hyperparameters governing the optimization procedure. Results are displayed in Figure 3.20. Both DGP variations outperformed the GP on prediction accuracy and uncertainty quantification (indicated by low CRPS), with my ESS-based DGP performing marginally better than the variational-inference-based competitor. It may be possible to improve the results from `gpytorch` with additional tuning, but it is noteworthy that my `deepgp` package implementation does not require any fine tuning (I utilized all the package defaults). See Section 4.3 for further discussion of competing DGP methodologies.

Encouraged by these results, I employed a contour location sequential design utilizing the DGP ESS surrogate with tricands and Pareto front acquisitions. I specified a failure contour at LD $= 3$ such that $f(x) < 3$ represents a failure (low efficiency). I started with a random LHS of size $n_0 = 100$ and acquired 400 points for a resulting design of size $n = 500$. Sensitivity and specificity (higher is better) for 10 MC repetitions (with re-randomized initial

LHS) are shown in Figure 3.21. The left panels show progress across the sequential designs, and the right panels show the results from static designs of equivalent size (copied from Figure 3.20 for reference). The DGP-tricands-Pareto sequential designs outperformed the static fits with as few as 200 evaluations of the simulator. Performance saturates after several hundred acquisitions, which is a useful metric for determining a satisfactory "stopping point" for the sequential design.

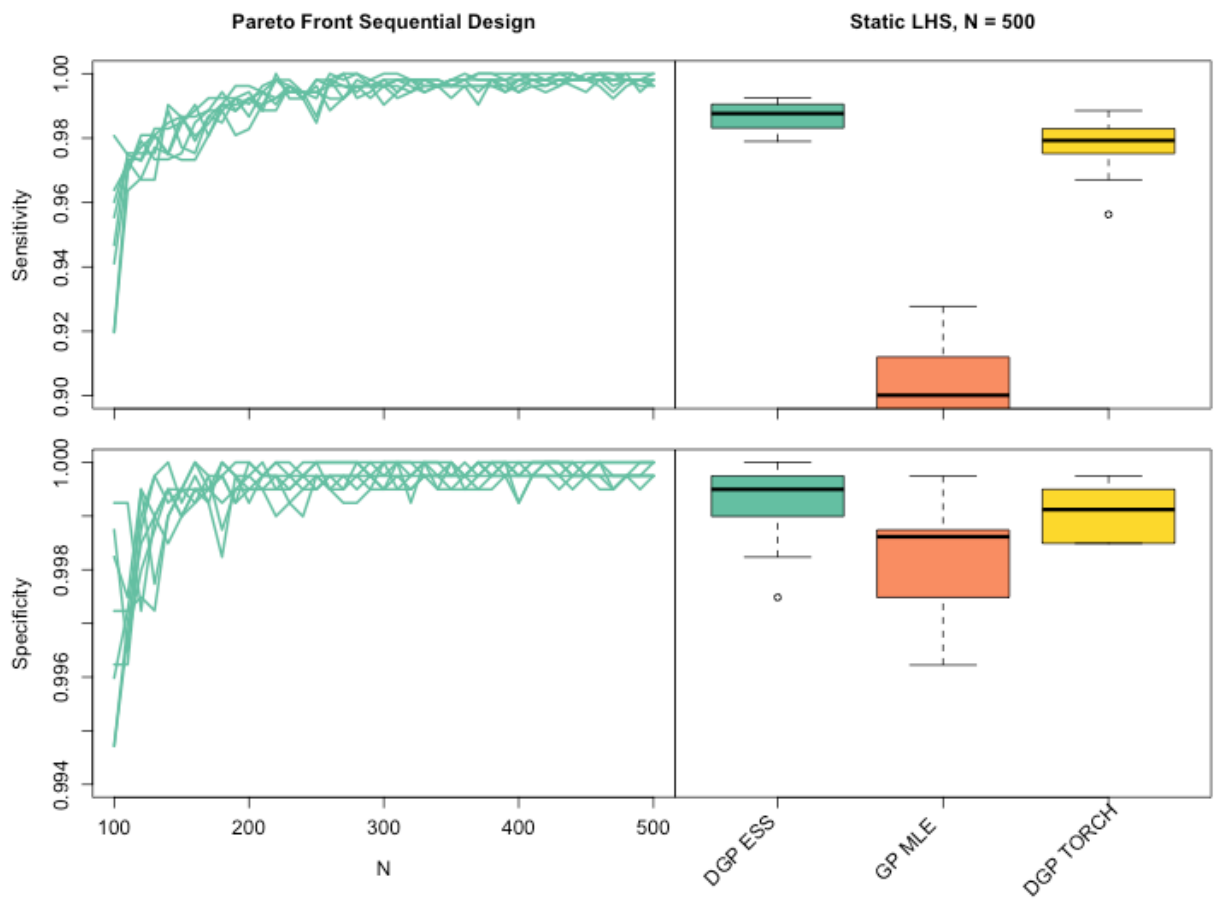


Figure 3.21: Sensitivity (top) and specificity (bottom) for "DGP ESS Pareto" sequential designs of the SU2 simulator (left panels). Right panels show results from static LHS designs.

Figure 3.22 shows one of the resulting sequential designs, visualized as a projection over the two most impactful inputs: angle of attack and mach number (speed). Although the designs are in seven dimensions, this low dimensional projection still provides some insights. Pareto

front acquisitions (filled circles and triangles) occasionally explore, but overwhelmingly focus on low angles of attack. All "failures" occurred at low angles of attack and high speeds. This behavior makes sense, as this is where drag is generally high and lift is generally low. Acquisitions were often placed near this boundary in the upper left corner, an indication that the sequential design procedure is properly honing in on the contour.

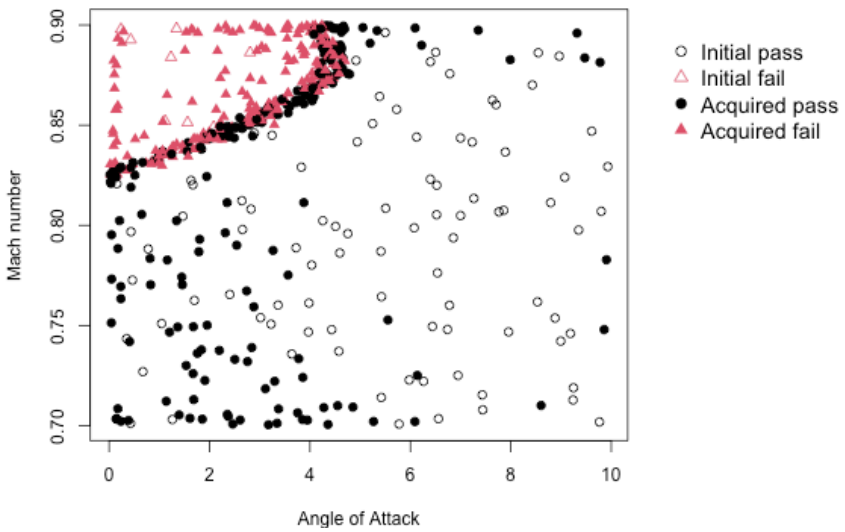


Figure 3.22: Two-dimensional projection of a "DGP ESS Pareto" sequential design for the 7-dimensional SU2 simulator.

## 3.4 Discussion

The non-stationary nature of DGPs makes them an excellent candidate for sequential design via active learning (AL) when simulation dynamics are characterized by stark regime shifts. The flexibility of a DGP, combined with an inferential scheme that offers thorough UQ through full posterior integration, allows for sequential designs that depart from traditional space-filling allocations and yield improved performance. I demonstrated DGP superiority over typical GP regression in examples ranging up to seven dimensions. I entertained three

unique AL objectives: variance reduction, Bayesian optimization, and contour location.

I was most surprised by how well a simple default setup – two layers with nodes matching the input dimension ($p = d$) – compared to more complex alternatives. Although there is evidence that even deeper GPs ($\geq 3$ latent layers) work well when there are abrupt regime shifts [20, 24], I was unable to identify any common computer simulation benchmarks which demanded that complexity. It may be that typical simulators simply aren't that pathological. Perhaps incorporating input connected networks [27], in which deep layers depend on both the previous layer and the original inputs, may increase the utility of deeper models. In higher dimensional settings, I found the relative success of two-layer DGPs with $p < d$ to be intriguing. My work was limited to post-hoc analysis of depth and node size; selecting these specifications in real-time, for example, would be of great practical interest.

Triangulation candidates proved useful in both the BO and CL settings, but the computational costs of the Delauney triangulation may become prohibitive in higher dimensions. In my experience, 8 dimensions is pushing the boundary of feasibility for the Delauney triangulation. There are some opportunities for improvement, such as only updating the portion of the triangulation near a newly acquired point. But I suspect that new methodology will be needed to replace the Delauney triangulation in higher dimension while still mimicing its behavior.

While I chose to use the Pareto front of entropy and posterior standard deviation for contour location, the utility of the Pareto front procedure is not limited to the entropy criterion nor to the contour location setting. Combining any acquisition criteria with posterior standard deviation and selecting acquisitions on the frontier has potential to encourage exploration while retaining exploitative behavior, which could be useful for a wide range of sequential design endeavors.

# Chapter 4

# Vecchia-approximated DGPs

Contrary to active learning set-ups, some computer experiments are computationally cheap, and large training datasets are easy to come by. When $n$ is large, cubic computational bottlenecks caused by the large matrix determinant and inverse computations of the GP likelihood (1.3) become prohibitive. These bottlenecks are exacerbated in a Bayesian DGP with dense covariance matrices at every Gaussian layer and many likelihood evaluations required for full posterior integration. A thrifty approximation is necessary. I introduced some popular GP approximations in Chapter 1, but I prefer Vecchia approximation [128].

The Vecchia approximation is motivated by computational bottlenecks in GP regression, which are compounded in a DGP setting. The underlying idea is basic: any joint distribution can be factored into a product of conditionals $p(y) = p(y_1)p(y_2 \mid y_1)p(y_3 \mid y_2, y_1) \cdots p(y_n \mid y_{n-1}, \ldots, y_1)$. This is true up to any re-indexing of the $y_i$'s. In particular, and to establish some notation for later, any joint likelihood (1.3) may be factored into a product of univariate likelihoods

$$\mathcal{L}(Y) = \prod_{i=1}^{n} \mathcal{L}\left(y_i \mid Y_{c(i)}\right) \tag{4.1}$$

where $c(1) = \emptyset$ and $c(i) = \{1, 2, \ldots, i-1\}$ for $i = 2, \ldots, n$. The Vecchia approximation instead takes a subset, $c(i) \subset \{1, 2, \ldots, i-1\}$, of size $|c(i)| = \min(m, i-1)$. When $m < n$, the strict equality of Eq. (4.1) is technically an approximation, yet I will use equality notation throughout when speaking of the general case with unspecified $m$. This approximation is

indexing-dependent for fixed $m < n$, but hold that thought for a moment. Crucially Eq. (4.1), in the context of Eqs. (1.3) and (1.1), induces a sparse precision matrix: $Q(X) = \Sigma(X)^{-1}$. The $(i,j)^{\text{th}}$ element of $Q(X)$ is 0 if and only if $y_i$ and $y_j$ are conditionally independent (i.e. $i \notin c(j)$ and $j \notin c(i)$). The Cholesky decomposition of the precision matrix, $U_x$ for $Q(X) = U_x U_x^\top$, is even sparser with fewer than $m$ off-diagonal non-zero entries in each row. I follow Katzfuss et al. [66] in working with the upper trianglular $U_x$, referred to as the "upper-lower" Cholesky decomposition.

A GP-Vecchia approximation requires two choices: an ordering of the data and selection of conditioning sets $c(i)$. There are many orderings that work well [47, 65, 121], but a simple random ordering is common [22, 122, 129]. The prevailing choice for conditioning sets is "nearest-neighbors" (NN) in which $c(i)$ comprises of integers indexing the closest observations to $x_i$ which appear earlier in the ordering. Approximations based on NN are sometimes referred to as NNGPs [22]. To demonstrate an ordering and NN conditioning set, the left panel of Figure 4.1 shows a grid of inputs with random ordering (the numbers plotted). [The other panels will be discussed in Section 4.1.3.] For point $i = 45$ (red triangle), the NN conditioning set of size $m = 10$ is highlighted by blue circles. These are the points closest to $x_{45}$ in Euclidean distance, with indices $j < i$ in the ordering. Sets $c(i)$ for all $i = 1, \ldots, 100$ are chosen similarly.

Under a GP, components of Eq. (4.1) are univariate Gaussian,

$$\begin{aligned} \mathcal{L}(y_i \mid Y_{c(i)}) \sim \mathcal{N}_1(\mu_i(X), \sigma_i^2(X)) \quad \text{where} \quad & B_i(X) = \Sigma(x_i, X_{c(i)})\Sigma(X_{c(i)})^{-1}, \\ & \mu_i(X) = B_i(X) Y_{c(i)}, \\ & \sigma_i^2(X) = \Sigma(x_i) - B_i(X)\Sigma(X_{c(i)}, x_i), \end{aligned} \tag{4.2}$$

and $X_{c(i)}$ is the row-combined matrix of $X$'s rows corresponding to indices $c(i)$. Foreshadowing a DGP application in Section 4.1, one may define $B_i(W)$, $\mu_i(W)$ and $\sigma_i^2(W)$ identically

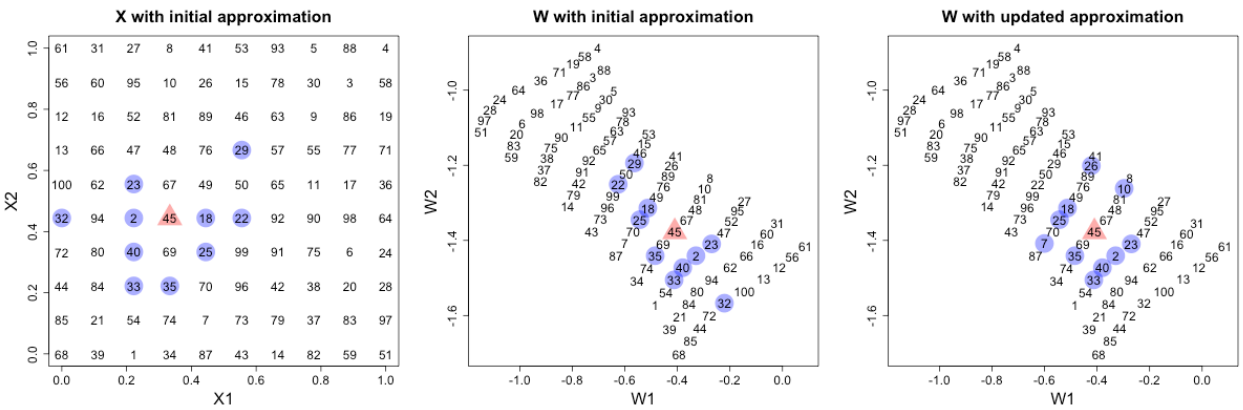


Figure 4.1: (Left) A uniformly spaced grid of inputs, randomly ordered. NN conditioning set for $x_{45}$ (red triangle) is highlighted by blue circles. (Middle) Same ordering/NN sets instead plotted as the "warped" $W$ from Figure 2.4. (Right) "Warped" $W$, but re-ordered randomly with NN conditioning adjusted accordingly.

but with $w/W$ in place of $x/X$. With this representation, I convert a large $n \times n$ matrix inversion ($\mathcal{O}(n^3)$) into $n$-many $m \times m$ matrix inversions ($\mathcal{O}(nm^3)$), a significant improvement if $m \ll n$.

The details of Vecchia-GPs, including numerous options for orderings, conditioning sets, hyperparameterizations, and computational considerations, are spread across multiple works [e.g., 21, 22, 33, 47, 48, 65, 66]. These specifications, along with software implementations [e.g., 34, 49, 68], can be rather complex. I do not need such hefty machinery in my DGP setup for computer surrogate models. Much of my research effort involved sifting through this literature to determine what is essential and which variations work best for DGP surrogates. As one example, latent layers $W$ and deterministic $Y = f(\cdot)$ utilize noise free modeling (small/zero nugget $g$), which affords several simplifications. In particular, I do not follow Datta et al. [22] and Katzfuss and Guinness [65] in distinguishing between noisy observations and latent "true" variables. This greatly streamlines development [Section 4.1] and reduces computational demands.

## 4.1 Posterior Inference

### 4.1.1 Inferential Building Blocks

I impose the Vecchia approximation at each layer of the DGP. Leveraging sparsity of the upper-lower Cholesky decomposition of the precision matrices, a two-layer Vecchia-DGP model may be represented as

$$Y \mid W \sim \mathcal{N}_n\left(0, (U_w U_w^\top)^{-1}\right) \qquad W_k \overset{\text{ind}}{\sim} \mathcal{N}_n\left(0, \left((U_x^{(k)})(U_x^{(k)})^\top\right)^{-1}\right), \quad k = 1, \ldots, p, \tag{4.3}$$

where $\Sigma(W)^{-1} = U_w U_w^\top$ and $\Sigma_k(X)^{-1} = (U_x^{(k)})(U_x^{(k)})^\top$. Each $W_k$, having its own Gaussian prior, *also* has its own Vecchia decomposition. Often these $U_x^{(k)}$ will share conditioning sets but have disparate hyperparameterization, e.g., unique lengthscales $\theta_x^{(k)}$. When $m = n$, this formulation is equivalent to Eq. (2.1). When $m < n$, $U_w$ and $U_x^{(k)}$ have induced sparsity.

I aim to conduct full posterior integration for this model, extending Eq. (2.2) to include integration over hyperparameters, but as I've mentioned several times, this is not analytically tractable. Posterior sampling via ESS and MH regarding $\mathcal{L}(Y \mid W)$ and $\mathcal{L}(W_k \mid X)$ requires three ingredients: (i) prior sampling, (ii) likelihood evaluation, and (iii) prediction at unobserved inputs. These are detailed here, for the model in Eq. (4.3), in turn. I shall focus on $\mathcal{L}(Y \mid W)$, but the idea is immediately extendable to $\mathcal{L}(W_k \mid X)$. Both are GPs, so the details only differ superficially in notation, and with iteration over $k = 1, \ldots, p$. Ultimately, these "building blocks" tie together to support posterior sampling, with details following in Section 4.1.2.

**(i) Prior.** Direct sampling of $Y^\star \sim \mathcal{N}_n(0, (U_w U_w^\top)^{-1})$, often called "conditional simulation", involves individually drawing $y_i^\star \sim \mathcal{N}_1(B_i(W) Y_{c(i)}^\star, \sigma_i^2(W))$ for $B_i(W)$ and $\sigma_i^2(W)$

defined with analogy to Eq. (4.2). An important difference, however, is that the application here crucially relies on *previously sampled* $Y^\star_{c(i)}$, meaning each $y_i^\star$ must be sampled *sequentially*. With an eye towards parallel implementation [Section 4.2], I instead leverage the sparsity of the upper-triangular $U_w$. Katzfuss and Guinness [65, Proposition 1] derived a closed-form solution for populating $U_w$, with $(j,i)^{\text{th}}$ entry

$$U_w^{ji} = \begin{cases} \frac{1}{\sigma_i(W)} & i = j \\ -\frac{1}{\sigma_i(W)} B_i(W)[\text{index of } j \in c(i)] & j \in c(i) \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

for $B_i(W)$ and $\sigma_i(W)$ in Eq. (4.2). With $U_w$ in hand, sampling $Y^\star$ follows Gelman et al. [38, App. A]:

$$Y^\star = (U_w^\top)^{-1} z \quad \text{where} \quad z \sim \mathcal{N}_n(0, \mathbb{I}). \tag{4.5}$$

Strategically, I avoid matrix inversions by using a forward solve of $U_w^\top Y^\star = z$.

**(ii) Likelihood.** Evaluations of $\mathcal{L}(Y \mid W)$ could similarly be calculated as the product of univariate Gaussian densities, combining Eqs. (4.1–4.2) via $\mathcal{L}(y_i \mid W) \sim \mathcal{N}_1(\mu_i(W), \sigma_i^2(W))$. I instead choose to leverage the sparse $U_w$ formulation (4.4), yielding the log likelihood

$$\begin{aligned} \log \mathcal{L}(Y \mid W) &\propto \log |(U_w U_w^\top)^{-1}|^{-1/2} - \frac{1}{2} Y^\top U_w U_w^\top Y \\ &\propto \sum_{i=1}^n \log(U_w^{ii}) - \frac{1}{2} Y^\top U_w U_w^\top Y, \end{aligned} \tag{4.6}$$

in which the sparse structure of $U_w$ allows for thrifty matrix multiplications.

**(iii) Prediction.** Given observed $Y \mid W \sim \mathcal{N}_n(0, (U_w U_w^\top)^{-1})$, i.e., training observations $Y$ and a burned-in ESS sample of $W$, I wish to predict $\mathcal{Y}$ for an $n_p \times d$ matrix of novel

$\mathcal{W}$. Note these novel $\mathcal{W}$ ultimately arise as samples following analogous application of the very same procedure I am about to describe, except for $W_k$ as the "response" drawn at novel testing sites $\mathcal{X}$ (more on this in Section 4.1.2). The simplest approach treats each row of $\mathcal{W}$ independently. Independent prediction is sufficient if only point-wise means and variances are required, as is common in many downstream surrogate modeling tasks. For each $i = 1, \ldots, n_p$, I form $c(i)$ with $m$ training locations from $W$ (details in Section 4.1.3). This imposes conditional independence among $\mathcal{Y}_i$ (i.e., $\mathcal{Y}_i$ is not conditioned on $\mathcal{Y}_j$ for $i \neq j$). The posterior predictive distribution then follows $\mathcal{Y}_i \sim \mathcal{N}_1(\mu_i(W), \sigma_i^2(W))$ with $\mu_i(W)$ and $\sigma_i^2(W)$ defined as in Eq. (4.2).

This independent treatment is fast and easily parallelized over index $i = 1, \ldots, n_p$. Consequently, it is the method I prefer for the benchmarking exercises of Sections 4.4–4.5, involving a cumbersome additional layer of Monte Carlo (MC) over training-testing partitions. Yet an imposition of independence among $\mathcal{Y}$ can be limiting. In some cases, joint prediction utilizing the full covariance structure $\mathcal{Y} \mid Y, W \sim \mathcal{N}_{n_p}(\mu^\star, \Sigma^\star)$ is essential. I can accommodate such settings as follows. First append $\mathcal{W}$ indices to the existing ordering of $W$, ensuring predictive locations are ordered *after* training locations, forming the full ordering $i = 1, \ldots, n, n+1, \ldots, n+n_p$. Conditioning sets $c(i)$ for $i = n+1, \ldots, n+p$ index *any* observations from $W$ or $\mathcal{W}$ with indices prior to $i$ in the combined ordering, thus allowing predictive outputs to potentially condition on other predictive outputs, in addition to nearby training data observations.

Next, "stack" training and testing responses in the usual way [42, Section 5.1.1]

$$\begin{bmatrix} Y \\ \mathcal{Y} \end{bmatrix} \sim \mathcal{N}_{n+n_p}\left(0, \quad \Sigma_{\text{stack}}\right) \quad \text{where} \quad \Sigma_{\text{stack}} = \Sigma\left(\begin{bmatrix} W \\ \mathcal{W} \end{bmatrix}\right) = \begin{bmatrix} \Sigma(W) & \Sigma(W, \mathcal{W}) \\ \Sigma(\mathcal{W}, W) & \Sigma(\mathcal{W}) \end{bmatrix}.$$

Then leverage Eq. (4.4) to analogously populate a "stacked" upper-lower Cholesky decom-

position,

$$U_{\text{stack}} = \begin{bmatrix} U_w & U_{w,\mathcal{W}} \\ 0 & U_{\mathcal{W}} \end{bmatrix} \quad \text{such that} \quad \Sigma_{\text{stack}} = \left(U_{\text{stack}} U_{\text{stack}}^\top\right)^{-1} = \left( \begin{bmatrix} U_w U_w^\top + U_{w,\mathcal{W}} U_{w,\mathcal{W}}^\top & U_{w,\mathcal{W}} U_{\mathcal{W}}^\top \\ U_{\mathcal{W}} U_{w,\mathcal{W}}^\top & U_{\mathcal{W}} U_{\mathcal{W}}^\top \end{bmatrix} \right)^{-1}.$$

An application of the partition matrix inverse identities (details in Appendix A.4–A.5) results in the following posterior predictive moments, after applying the usual multivariate normal conditioning identities for $\mathcal{Y} \mid Y, W$ (1.4):

$$\mathcal{Y} \mid Y, W \sim \mathcal{N}_{n_p}\left(\mu^\star, \Sigma^\star\right) \quad \text{for} \quad \mu^\star = -(U_{\mathcal{W}}^\top)^{-1} U_{w,\mathcal{W}}^\top Y \quad \text{and} \quad \Sigma^\star = \left(U_{\mathcal{W}} U_{\mathcal{W}}^\top\right)^{-1}. \tag{4.7}$$

These are simplified versions of the moments provided by Katzfuss et al. [66], thanks to a streamlined latent structure and the imposition that predictive locations must be ordered after training locations. Naturally, if no predictive locations condition on others then $U_{\mathcal{W}}$ and $\Sigma^\star$ will be diagonal, and I can return to the simpler implementation of independent predictions.

Katzfuss et al. [66] remark that conditioning on other predictive locations, i.e., using joint $\mu^*$ and $\Sigma^\star$, is more accurate than conditioning only on training data, say following the independent $\mu_i$ and $\sigma_i^2$ version I presented first. Anecdotally, in my own empirical work, I have found this difference to be inconsequential. Unless a joint $\Sigma^\star$ is required, say for calibration [70], I prefer the faster, parallelizable, independent approach. [Both are provided by my software; more in Section 4.2.] In settings where it might be desirable to reveal/leverage posterior predictive correlation, but perhaps it is too computationally burdensome to work with $n_p^2$ pairs of testing sights simultaneously, a hybrid or batched scheme might be preferred.

### 4.1.2 Inferential Scheme

Building blocks (i–iii) in hand, posterior sampling by MCMC may commence following Section 2.2. In other words, the underlying inferential framework is unchanged modulo an efficient (Vecchia) method for (i) prior sampling, (2) likelihood evaluation, and (3) prediction. Rather than duplicate details here, allow me point out a few relevant highlights. In model training, every evaluation of a Gaussian likelihood utilizes Eq. (4.6), whether for inner ($W_k$) or outer ($Y$), matched with $U_x^{(k)}$ and $U_w$, respectively, and with appropriate covariance hyperparameters, e.g., $\theta_x^{(k)}$, embedded into the $B_i(\cdot)$ and $\sigma_i(\cdot)$ components of said $U$ matrices (4.4). When employing ESS for $W_k$, say, random samples from the prior follow Eq. (4.5). The MCMC scheme remains unchanged in its structure, while every under-the-hood calculation is replaced with its Vecchia-approximated counterpart.

To predict at unobserved inputs $\mathcal{X}$, i.e., Section 2.3, replace traditional GP prediction at each Gaussian layer (1.4) with its Vecchia counterpart (4.7). For each candidate (burned-in/thinned) MCMC iteration, predictive locations $\mathcal{X}$ are mapped to "warped" locations $\mathcal{W}$, which are then mapped to posterior moments for $\mathcal{Y}$, with each step following Eq. (4.7). The resulting moments are post-processed, with ergodic averages yielding the final posterior predictive moments. Joint predictive distributions may be replaced with independent pointwise, and parallelized predictions as described in Section 4.1.1 in the presense of a large/dense testing set. Note, a DGP predictive distribution is not strictly Gaussian, even though it arises as an integral over Gaussians. However, I find that ergodic averages, represented abstractly here by empirical moments $\bar{\mu}$ and $\bar{\Sigma} + \mathbb{C}\mathrm{ov}(\mu)$ through the law of total variance (2.13), are a sufficient substitute for retaining thousands of high-dimensional MCMC draws of $\mu^\star$ and $\Sigma^\star$, say, or their pointwise analogues.

It is important to briefly acknowledge the substantial computation inherent in this inferential

scheme. The requisite MCMC requires thousands of iterations, each of which necessitates multiple likelihood (4.6) evaluations. Predictions require averaging across these draws, although thinning can reduce this effort. Despite these hefty computational demands, efficient parallelization (described in more detail momentarily), strategic initializations of latent layers, and other sensible pre-processing yield feasible compute times even with large data sizes. For example, the Vecchia-DGPs of Section 4.5 with $n = 100,000$ may be fit in less than 24 hours on a 16-core machine. I aim to show that this investment pays dividends compared to faster GP and DGP alternatives in terms of prediction accuracy and UQ [Section 4.4–4.5].

### 4.1.3 Ordering and Conditioning

Each Gaussian component of the DGP could potentially have its own ordering and conditioning set, $c_x^{(k)}(i)$ for $U_x^{(k)}$ and $c_w(i)$ for $U_w$ in the two-layer model, in which orderings denoted by $i$ need not be the same. Since each $c_x^{(k)}(i)$ acts on the same input space, I simplify the approximation by sharing ordering and conditioning sets across $k = 1, \ldots, p$, resulting in only two orderings and two conditioning sets, $c_x(i)$ and $c_w(j)$. Here, separate indices $i$ and $j$ are intended to convey uniqueness.

These choices are not part of the stochastic process describing the data generating mechanism, although that is an interesting possibility I discuss in Section 4.6. A consequence of this is that once a chain has been initialized under a particular ordering, yielding $U_x$ or $U_w$ up to hyperparameters like $\theta$ which *are* included in the hierarchy describing the stochastic process, $c_x(i)$ and $c_w(j)$ must remain fixed throughout the MCMC in order to maintain detailed balance. It occurred to me to try randomizing over orderings from one MCMC iteration to the next, but the chain does not burn in/achieve stationarity. Each change to one of $c_x(i)$ or $c_w(j)$ causes the chain to "jump" somewhere else. Nevertheless, it could be advantageous to

customize aspects of a Vecchia ordering and conditioning dynamically, say based on a DGP fit or other analysis, or to hedge by averaging results from multiple orderings. This is fine with independent chains.

With this in mind, I adopt the following setup. Begin with a random ordering of indices and subsequent NN conditioning. This follows the recommendation of Guinness [47] and mirrors other recent work on NNGPs [22, 122, 129]. Then select conditioning sets based on NN, as eponymous in the NNGP/Vecchia literature. In $X$-space, calculating $c_x(i)$ as the $\min(m, i-1)$ nearest points (of lower index) is straightforward. These locations are anchored in place by the experimental design. For latent Gaussian layer $W$, NN conditioning sets can be more involved. Since $W$ is unobserved, I start with no working knowledge of the nature of the warping. [My prior is mean-zero Gaussian under a distance-based covariance structure with unknown hyperparameterization.] A good automatic initialization for the MCMC is to assume no warping (i.e. $W = X$). In that setting, NN for $W$ based on relative Euclidean distance in $X$ space is sensible.

Such a conditioning set is even workable after considerable posterior sampling, whereby $W$ may have diverged from the identity mapping with $X$. I find that in practice each individual nodes' (i.e., $W_k$) contribution to the overall multidimensional warping for $k = 1, \ldots, p$ is usually convex. As a visual, consider again Figure 4.1. The right two panels show two different orderings (both random) and conditioning sets (both NN in a certain sense) for a $W$ arising as a function of $X$ corresponding to the maps in Figure 2.4, only now visualized as spread of $W_1$ and $W_2$ in two-dimensional space. The locations of the observations (marked by numbers) represent a warping of the original evenly-gridded inputs (left panel). The middle panel shows the conditioning set $c_w(45)$ that was selected based on NN in $X$ space. Observe that these highlighted points are equivalent to those of the left panel.

Selecting $c_w(j)$ based on $X$ is a good starting point, but I envision scope to be more strate-

gic. Given posterior information about $W$, one may wish to update $c_w(j)$ in light of that warping. For example, NN on $W$ could be calculated after burn-in and used as the basis of $U_w$ conditioning for a re-started chain. Such an operation could be viewed as a nonlinear extension of sensitivity pre-warping [130], tailored to the Vecchia approximation. The right panel of Figure 4.1 shows what such re-conditioning might look like under a novel random reordering. There is some precedence for evolving neighborhood sets in this way from the ordinary Vecchia-GP literature. For example, Katzfuss et al. [67] use estimated separable lengthscale parameters to find NN based on re-scaled inputs $X/\sqrt{\theta}$, vectorizing over columns; Kang and Katzfuss [62] extend that to full kernel/correlation based re-scaling. Both are situated in an optimization based inferential apparatus, and the authors describe a careful "epoch-oriented" scheme to circumvent convergence issues, analogous to maintaining detailed balance in MCMC. My particular re-burn-in instantiation of this idea, described above, represents a natural extension: an affine warping of inputs for NN calculations is upgraded to a nonlinear one via latent Gaussian layers. Yet in my empirical work exploring this idea [Section 4.4], I disappointingly find little additional value realized by the extra effort for DGPs. I speculate in Section 4.6 that this may be because I haven't yet encountered any applications demanding highly non-convex $W$.

A final consideration involves extending orderings and neighborhood sets to testing sites when sampling from the posterior predictive distribution. Here, I again use NN sets to select $c_x(i)$ and $c_w(j)$ for predictive/testing locations $\mathcal{X}$ (which are mapped to $\mathcal{W}$). In $X$ space, NN sets are fixed once for each row of $\mathcal{X}$. In $W$ space, after MCMC sampling (i.e., model "training"), I leverage the learned warpings ($W^{(t)}$ for $t \in \mathcal{T}$, after discarding burn-in and thinning as desired) and calculate NNs in that space to maximize the efficacy of the approximation at each location. Specifically, for each row of $\mathcal{W}$, I re-calculate "warped" NN sets for each sample $W^{(t)}$. Resulting predictions are combined with expectation taken over

all $t \in \mathcal{T}$. This re-calculation of $c_w(i)$ for each $t$ requires extra effort, but it is not onerous and focuses computation where it is most needed, in making the best possible prediction for each testing location.

## 4.2 Implementation Details

I provide an open-source implementation as an update to the `deepgp` package [110] for R on CRAN. Although I embrace a bare-bones approach, my R/C++ implementations of the "building blocks" in Section 4.1.1 are heavily inspired by the more extensive `GPvecchia` [68] and `GpGp` [49] packages. Computational speed relies on strategic parallelization and careful consideration of sparse matrices. For example, I utilize `OpenMP` pragmas to parallelize the calculation of each row of the sparse $U_w$ and $U_x^{(k)}$ (4.4). I use `RcppArmadillo` [29] in C++ and `Matrix` [7] in base R to handle sparse matrix calculations, aspects of which are also parallelized (but usually to a lesser degree) under-the-hood.

While my Vecchia-DGP implementation in `deepgp` is distinct from its fully (un-approximated) counterpart, they share an interface for ease of use. A `vecchia` indicator to the existing `fit` functions triggers approximate inference, i.e.,

```
R> fit <- fit_two_layer(X, Y, vecchia = TRUE, m = 25, true_g = eps)
R> fit <- predict(fit, Xpred, lite = TRUE, m = 25)
```

The neighborhood size is specified as `m` $= m = \min\{25, n-1\}$ by default, where choosing $n-1$ results in no approximation, but still uses the Vecchia implementation, which is useful for debugging and benchmarking. Notice training and prediction accommodate distinct $m$. I have found that $m = 25$ (for both) works well in low/no-noise settings, but it may be advantageous to use larger $m$ for prediction or to scale $m$ with $n$. I additionally allow

`predict` calls to toggle between independent (`lite = TRUE`) or joint predictions (`lite = FALSE`).

A two-phase MCMC option, updating orderings and NN conditioning sets based on a burned-in warping [Section 4.1.3] is supported by providing `re_approx = TRUE` to `continue`, an S3 method automating additional MCMC from the end of a previous chain. The `true_g` argument is optional. If it is not provided, then a nugget is estimated along with other hyperparameters. In my experiments later, I fix `eps = 1e-8` for all but the satellite drag example [Section 4.5] where I follow others in using `eps = 1e-4`.

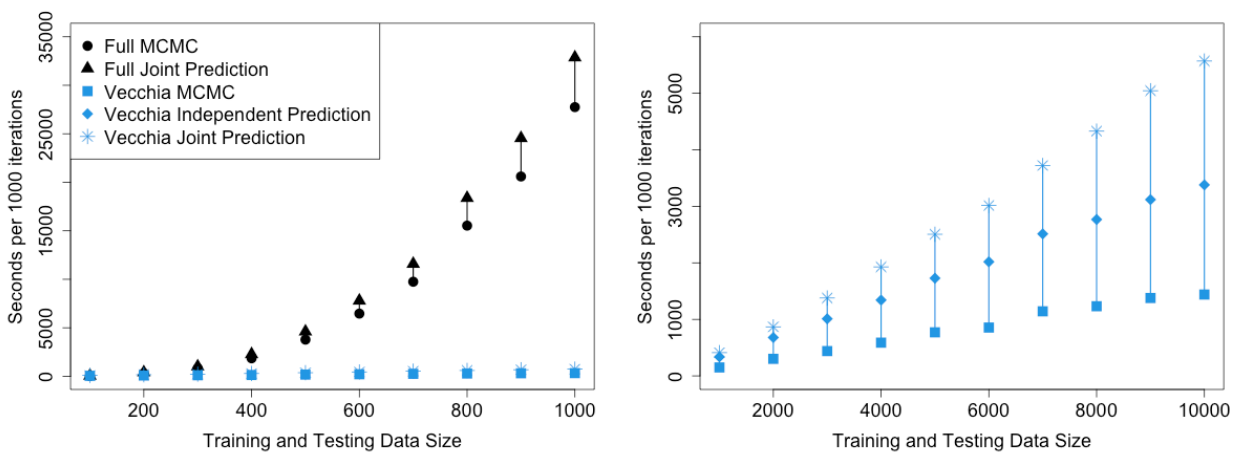


Figure 4.2: Computation time in seconds per 1000 MCMC iterations for both training and prediction (connected with vertical lines) of a two-layer DGP with and without Vecchia approximation ($m = 25$).

To demonstrate computational improvements over the full un-approximated implementation, the left panel of Figure 4.2 compares the computation time of 1000 such MCMC iterations between my full implementation (black) and my proposed Vecchia-DGP (blue). A 16-core, hyperthreaded, Intel i9 CPU at 3.6GHz, was used to collect these timings. The full, non-Vecchia implementation experiences cubic-in-$n$ costs and is not generally feasible for sample sizes above several hundred. The Vecchia implementation scales linearly-in-$n$, allowing for much larger data sizes. To contrast approaches to prediction outlined in Section 4.1.1, the

right panel of Figure 4.2 shows computation times for Vecchia-DGPs on larger training data sizes with both independent (diamonds) and joint (stars) schemes. Observe that independent predictions scale linearly in both $n$ and $n_p$, but joint predictions are more costly. Note the change in scale of the $y$-axes from the left to the right panel; both independent and joint predictions are leagues faster than un-approximated counterparts.

## 4.3 Competing Methodology and Software

As an alternative to MCMC, others have embraced variational inference (VI) in which the intractable DGP posterior (2.2) is approximated with a simpler family of distributions, which are often also Gaussian [20]. Inspired by deep neural networks, Bui et al. [13] proposed an Expectation Propogation (EP) scheme for DGPs, which is closely related to VI. Salimbeni and Deisenroth [106] broadened previous VI-like approaches for DGPs by allowing intra-layer dependencies, naming their method "doubly stochastic variational inference" (DSVI). The main advantage of these approaches is that integration is replaced by optimization, which requires less work. The disadvantage is that optimization ignores uncertainty; the fidelity of a VI approximation is linked to the choice of variational family, rather than directly to computational effort. More MCMC always improves posterior resolution; more VI does not. Hyperparameters don't neatly fit into variational families otherwise preferred for latent nodes, so they often get ignored, or their tuning is left to external validation schemes. By contrast, extra Metropolis easily accommodates a few more hyperparameters without hassle.

In order to handle training data sizes ($n$) upwards of hundreds of thousands, VI-based DGPs utilize *inducing points*, an umbrella term covering ideas developed separately as *pseudo-inputs* in machine learning [e.g., 117] and as *predictive processes* in geostatistics [e.g., 4]. Inducing points impose a low-rank kernel structure by measuring distance-based correlations through

a smaller subset of $m \ll n$ reference locations or "knots" in $d$-dimensions. Woodbury matrix identities improve decomposition of the implied full $n \times n$ structure from $\mathcal{O}(n^3)$ to $\mathcal{O}(nm^2)$. Although often framed as an "approximation", inducing points can represent a fundamental change to the underling kernel structure. Large $n$ and $m$ small enough to sufficiently speed up calculations can result in low-fidelity or "blurry" GP approximations [129]. Moreover, optimizing inducing point placement can be fraught with challenges [e.g., 37]. DSVI uses $k$-means to place inducing points near clusters of inputs, but computer experiments often deploy space-filling designs which would ensure there are no clusters.

The Vecchia approximation offers an alternative to inducing points, but without introducing auxiliary quantities. Although it is sometimes cast as a novel modeling framework rather than an approximation [22], a key advantage is that it doesn't fundamentally change the underlying kernel structure – at least not in the way inducing points do. Rather, it more subtly imposes sparsity in its inverse Cholesky factor. Although Vecchia can be higher on the computational ladder ($\mathcal{O}(nm^3)$), it is able to provide good approximations with $m$ much smaller than that required of inducing points without the "blur" or hassle in tuning the locations of $m \times d$ quantities. Wu et al. [129] entertain Vecchia in lieu of inducing points for ordinary GPs via VI with favorable results. It may only be a matter of time before Vecchia is deployed with VI for DGPs. I prefer MCMC for its UQ properties.

Other alternatives to inducing points in a VI context have been suggested, including random feature expansion [RFE; 74]. Extending RFE from ordinary to DGPs has been the subject of several recent papers [18, 73, 81], with some success. Others have taken the opposite route, keeping inducing points but swapping out VI for Hamiltonian Monte Carlo [HMC; 9] for DGPs [51]. HMC has an advantage over VI in that hyperparameters can easily be subsumed into the inferential apparatus without external validation. I show [Section 4.4] that this leads to performance gains on predictive accuracy in surrogate modeling settings.

Nevertheless, DSVI is generally considered the state-of-the-art inferential method for DGPs in machine learning. I think this is largely to do with computation and prowess in classification tasks. DSVI's inducing point approximation enables mini-batching to massively distribute a stochastic gradient descent. This seems to work well in classification settings where resolution drawbacks are less acute, but my experience [Section 4.4] suggests this may not extend to the low-noise regression settings encountered in surrogate modeling of computer simulations.

Ultimately, benchmarking against such alternatives comes down to software, as even the best methodological ideas are only as good, in practice, as their implementations. DSVI is neatly packaged in `gpflux` for Python [26], but requires specifying hyperparameters. Default settings were not ideal for my test problems, and I found manual tuning to be cumbersome. The sampling-based HMC implementation is available on the authors' GitHub page [51]. This performs better, I think, precisely because of its ability to more automatically tune hyperparameters in an Empirical Bayes/EM fashion, but uncertainty in these is not included in the posterior predictions. RFE software is on the authors' GitHub page [18], but relies on the specification of myriad inputs, with few defaults provided. Despite attempts to port example uses to my surrogate modeling setting, I had limited success. I found that results were uniformly inferior to DSVI, and no predictive uncertainties were provided leading me to ultimately drop RFE from further consideration. Finally, EP is available on the authors' GitHub page [13], but was written in Python2 and relies on legacy versions of several dependencies; I was unable to reproduce a suitable environment to try their code.

## 4.4 Synthetic Experiments

I include the following comparators:

- DGP VEC: my Vecchia-DGP, via `deepgp` using defaults, Matèrn $\nu = 5/2$ kernel, independent predictions, and "warped" conditioning sets. See Section 4.2.

- DGP FULL: full, un-approximated analog of DGP VEC, with everything otherwise identical (also via `deepgp`). This comparator was not feasible for all data sizes.

- DGP DSVI: from Salimbeni and Deisenroth [106], implemented in `gpflux`, with Matèrn $\nu = 5/2$ kernel. I follow package examples in using 100 $k$-means located inducing points. For numerical stability I required `eps = 1e-4`, lower bounding the noise parameter.

- DGP HMC: from Havasi et al. [51], again using 100 inducing points. This code only supports a squared exponential kernel and estimates (i.e., does not fix) the noise parameter ($g$/`eps`). I found no easy way to adjust these specifications, so I let them be.

- GP SVEC: scaled Vecchia GP of Katzfuss et al. [67], via `GPveccia` and `GpGp`. This is a fast "shallow" GP where kernel hyperparameters are estimated (4.6) via lengthscale-adjusted conditioning sets. I use `m = 25`, Matérn $\nu = 5/2$ kernel, and independent predictions to match DGP VEC.

All DGP variations are restricted to two layers. My metrics include out-of-sample root mean squared error (RMSE) and continuous rank probability score [CRPS; 39]. Lower is better for both. While RMSE focuses on accuracy of predictive means, CRPS incorporates point-wise predictive variances, thus providing insight into UQ. Although my DGP VEC is able to provide full predictive covariance, my competitors DGP DSVI, DGP HMC, and GP SVEC cannot.

Code to re-produce all results (including for competitors) is available on my public GitHub

repository.[1]

**Schaffer Function.** The two-dimensional "fourth" Schaffer function can be found in the Virtual Library of Simulation Experiments [VLSE; 124]. I follow the second variation therein,

$$f(x_1, x_2) = 0.5 + \frac{\cos^2\left(\sin(|x_1^2 - x_2^2|)\right) - 0.5}{\left[1 + 0.001\left(x_1^2 + x_2^2\right)\right]^2} \quad \text{for} \quad x_1, x_2 \in [-2, 2].$$

The function is characterized by steep curved inclines followed by immediate drops. These quick turns are challenging for stationary GPs, making the Schaffer function an excellent candidate for DGPs. I fit models to Latin hypercube samples [LHS; 85] of training sizes $n \in \{100, 500, 1000\}$ with fixed noise $g = 10^{-8}$. I use LHS testing sets of size $n_p = 500$. Results for 20 MC repetitions – all stochastic components from training/testing re-randomized – are displayed in Figure 4.3.

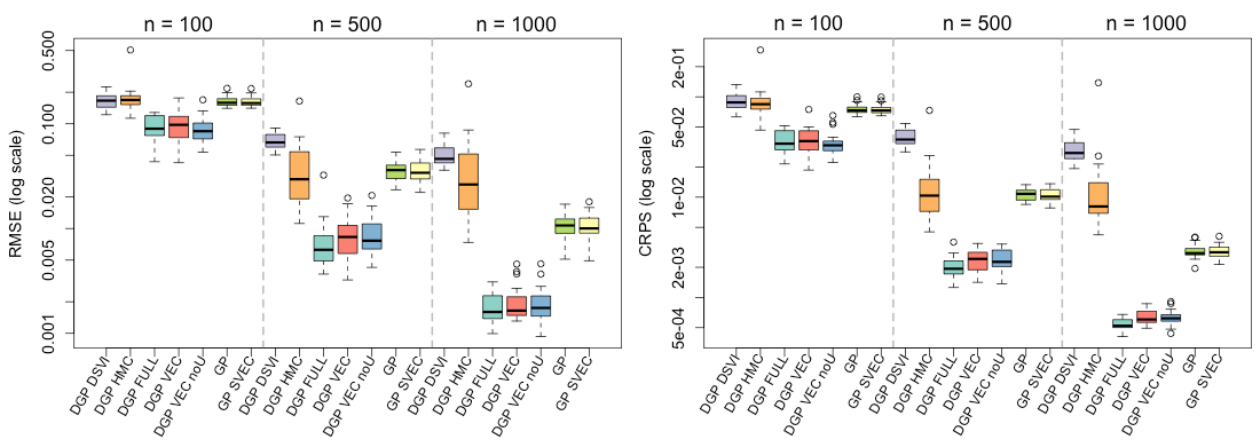


Figure 4.3: RMSE (left) and CRPS (right) on log scales for fits to the two-dimensional Schaffer function as training size ($n$) increases. Boxplots represent the spread of 20 MC repetitions.

All variations of my DGP fits outperform competitors by both metrics. Crucially, my Vecchia-DGP (DGP VEC) matches the performance of the un-approximated DGP (DGP

[1] https://bitbucket.org/gramacylab/deepgp-ex/

FULL). For this example I additionally implemented a "DGP VEC noU" comparator, identical to DGP VEC but without ordering/conditioning sets reset after pre-burn-in. Observe that this additional work is of dubious benefit empirically. Aside from MC variability, DGP VEC matches DGP VEC noU. Going forward I shall drop DGP VEC noU, focusing on my preferred DGP VEC setup, without evidence that results are better or worse and despite the additional (marginal) cost. Additional discussion is deferred to Section 4.6. Finally, at the outset I expected the stationary GP (GP SVEC) to perform poorly given the complexity of the response surface, but was surprised to see GP SVEC holding its own against DGP HMC, and eventually surpassing it with $n = 1000$. I suspect this is a consequence of "blurry" inducing point approximations.

To investigate the choice of conditioning set size $m$ for the Vecchia-DGP (DGP VEC) and scaled Vecchia-GP (GP SVEC), I recreated this exercise with $n = 1,000$ and varied $m \in \{5, 10, 25, 50, 100\}$. Resulting RMSE and CRPS from 20 MC repetitions are shown in Figure 4.4.

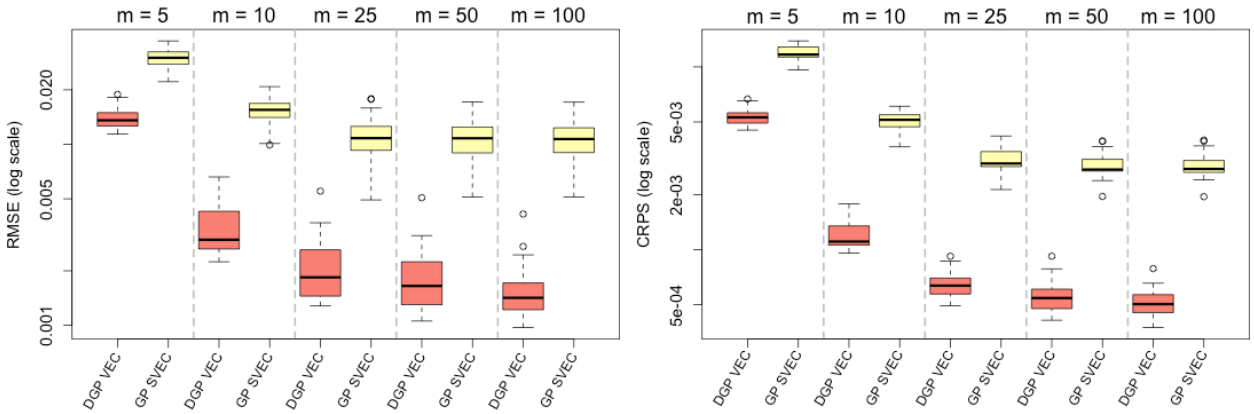


Figure 4.4: RMSE (left) and CRPS (right) for fits to the 2d Schaffer function as conditioning set size ($m$) increases (same $m$ used for training and prediction). Boxplots represent the spread of 20 repetitions.

As expected, increasing the size of the conditioning set improves predictive accuracy for both models. However, the benefits of conditioning on more points diminishes beyond $m = 25$.

While present in both models, this "leveling off" effect appears more starkly in the stationary GP SVEC than the non-stationary DGP VEC.

**G-function.** The G-function (2.14), also in the VLSE, is defined in arbitrary dimension. I worked with $d = 2$ in Figures 2.4 and 4.1. Here, I expand to $d = 4$. [I provided a preview of these results in Figure 2.5.] Higher dimensionality raises modeling challenges and demands larger training sets. I fit models to LHS samples of training sizes $n \in \{3000, 5000, 7000\}$ with fixed noise $g = 10^{-8}$. LHS testing sets were of size $n_p = 5000$. Results for 20 MC repetitions are displayed in Figure 4.5.

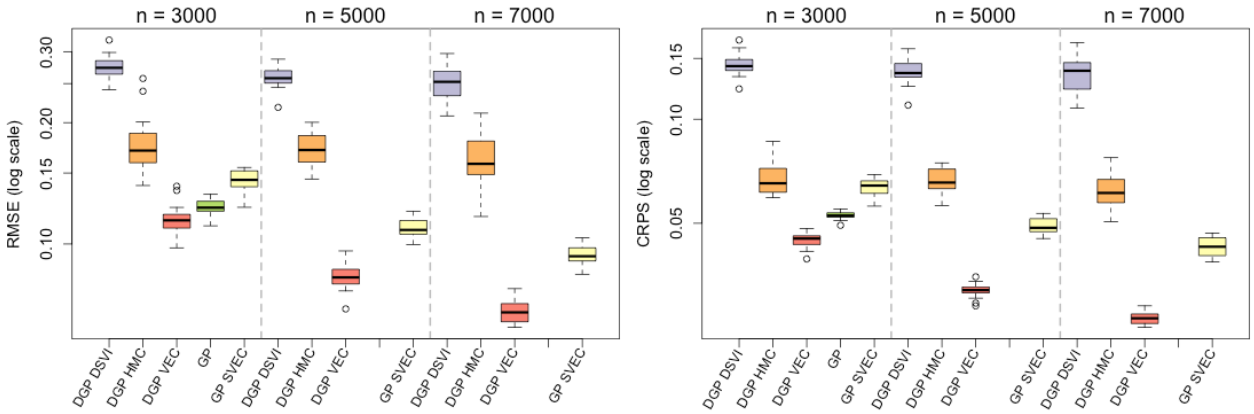


Figure 4.5: RMSE (left) and CRPS (right) on log scales for the four-dimensional G-function.

Again, my DGP VEC outperforms both deep and shallow competitors. DGP HMC, aided by its ability to estimate hyperparameters, bests DGP DSVI, but neither benefits from additional training data. Their predictive capability appears to saturate; I suspect this is due to inducing point approximations. While it is possible to increase the number of inducing points and potentially improve things, this requires adjustments to the source code and, in my experience, yields only marginal improvements before computation becomes prohibitive. GP SVEC is able to adapt to larger training sizes and surpass these deeper models. My DGP VEC benefits from all of these: estimation of hyperparameters, additional depth, and

learning from additional training data.

**G-function with Noise.** As an example of a noisy simulation, I re-create the Monte Carlo exercise for the G-function [83] with the addition of additive Gaussian noise $\epsilon_i \sim \mathcal{N}(0, 0.01^2)$. The set-up is equivalent to that of Section 4.4, except that each model is now tasked with estimating a noise parameter (i.e. `true_g = NULL` in `deepgp`). The DGP HMC and GP SVEC models have built-in capability to estimate noise. The DGP DSVI model does not, so I simply fix the noise parameter to the true variance ($g = 0.01$). To account for the extra challenge of distinguishing signal from noise, I double the data sizes to $n \in \{6000, 10000, 14000\}$ and $n_p = 10000$.

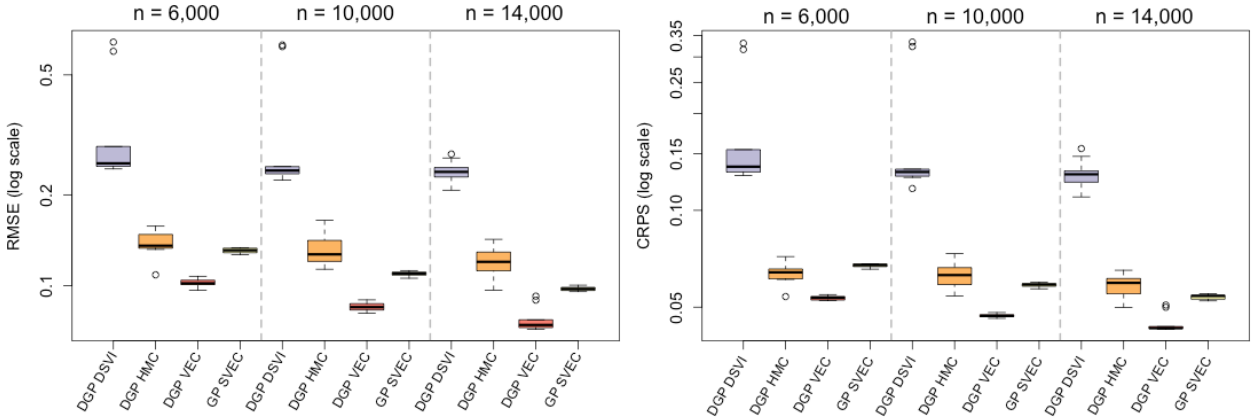


Figure 4.6: RMSE (left) and CRPS (right) on log scales for the four-dimensional G-function observed with Gaussian white noise. Boxplots represent the spread of 10 MC repetitions.

The results resemble Figure 4.5, suggesting the addition of noise does not affect the comparative efficacy of the models. DGP HMC and GP SVEC perform similarly, the former benefiting from the flexibility of DGP layers and the latter benefiting from the Vecchia approximation (as compared to inducing points). The DGP VEC model outperforms across the board. When matched by training/testing data, DGP VEC had lower RMSE and CRPS than each of these comparators in 30/30 trials.

**G-function in Higher Dimension.** To display functionality in higher dimension, I recreate the noise-free MC exercise for the G-function, this time expanding to $d = 6$. Higher dimension demands larger data sizes; I entertain random LHS training designs of size $n \in \{10000, 20000, 30000\}$ and LHS testing designs of size $n_p = 20000$. Results are shown in Figure 4.7.

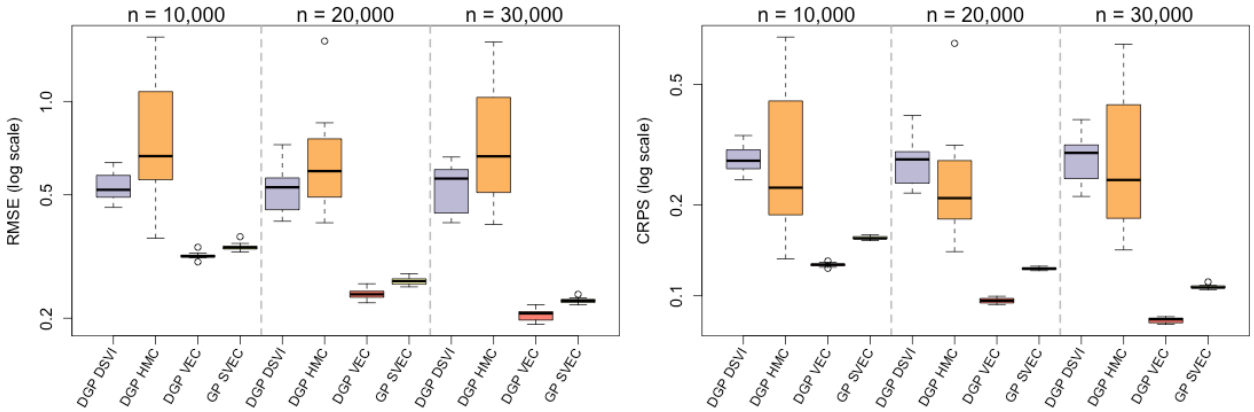


Figure 4.7: RMSE (left) and CRPS (right) on log scales for the 6d G-function. Boxplots represent the spread of 10 MC repetitions.

As in the lower dimensional exercises, DGP VEC outperforms on both prediction error and uncertainty quantification. Both DGP DSVI and DGP HMC are limited by inducing point approximations which are especially blurry in high dimensions. I acknowledge that the DGP HMC model is at a slight disadvantage since it estimates a noise parameter – the poor fits from this model may be over-estimating the noise.

## 4.5 Satellite Drag Computer Experiment

The *Test Particle Monte Carlo (TPM)* simulator [86] models the bombardment of satellites in low earth orbit by atmospheric particles; TPM returns coefficients of satellite drag based on particle composition and seven input variables specifying the orientation of the satellite.

I worked with this simulator on a restricted domain in Section 3.1.5. Researchers at Los Alamos National Lab, who developed the TPM library, wished to build a surrogate achieving less than 1% prediction error (measured in root mean squared percentage error, RMSPE) across the entire domain with as few runs of the simulator as possible. Sun et al. [123] used locally approximated GPs to reach the 1% goal with one million training data points. Katzfuss et al. [67] were later able to reach lower RMSPE's using scaled Vecchia GPs (GP SVEC). Here, I show that my Vecchia-DGP is able to beat the 1% RMSPE benchmark with as few as $n = 50,000$ (and can beat it consistently with $n = 100,000$), and provide better UQ than the stationary GP SVEC alternative.

I work specifically with the GRACE satellite, specified by a seven-dimensional input configuration, and a pure Helium atmospheric composition. For training data, I use random samples from Sun et al. [123]'s one million runs in sizes of $n \in \{10000, 50000, 100000\}$. I use random out-of-sample testing sets of size $n_p = 50,000$ from the complement, and follow Sun et al. [123] in fixing $g = 10^{-4}$. TPM simulations are technically stochastic, but the noise is very small.

In light of these large training data sizes and the accompanying computational burden, I make some strategic choices to initialize my DGP models and set up the MCMC for faster burn-in. First, I scale the seven input variables using estimated vectorized length-scales from GP SVEC (e.g., $X_i/\sqrt{\theta_i}$). This "pre-scaling" is common in computer surrogate modeling [e.g., 130], and mirrors the "scaled" component of the GP SVEC model. Second, I "seed" my MCMC by first running a long, thoroughly burned-in, set of iterations for one sample of $n$ = 10,000 and use the burned-in samples from this fit to initialize the chains for the larger data sets. This isn't necessary in practice but helps reduce the burden of repeated applications in a MC benchmarking context.

Results for 10 MC repetitions are displayed in Figure 4.8. Observe that DGP DSVI and

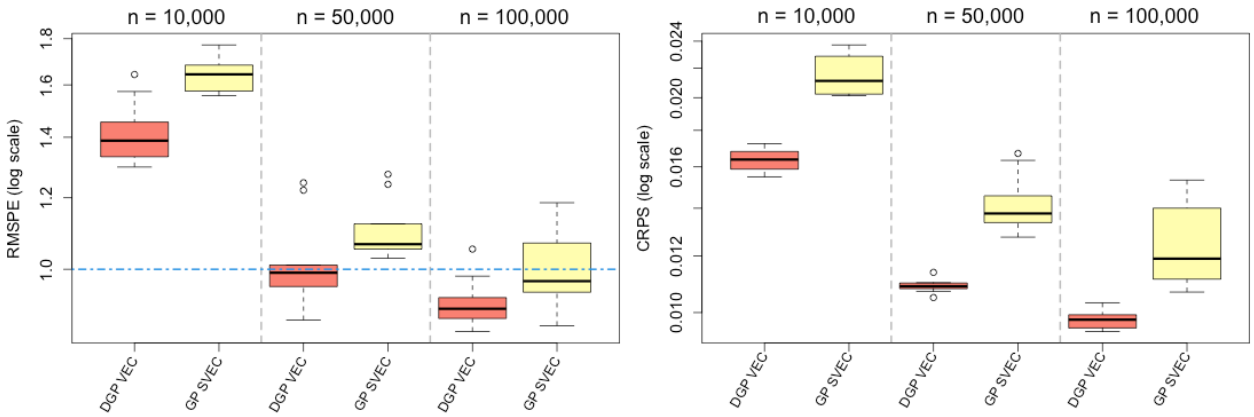


Figure 4.8: RMSE (left) and CRPS (right) on log scales for fits to the satellite drag simulation. DGP DSVI and DGP HMC are omitted (with RMSPE's betIen 30-35%). Horizontal line marks 1% RMSPE goal.

DGP HMC results are omitted from these plots. Because they were not competitive (each producing RMSPE's between 30-35%), their inclusion would severely expand the $y$-axis of the plots and render them hard to read. My DGP VEC consistently outperforms the shallow/stationary GP SVEC and is able to achieve the 1% RMSPE goal with as few as 50,000 training observations. Compared to GP SVEC counterparts (with matched training/testing data), DGP VEC models have lower CRPS in all 30 MC repetitions (represented by 3 boxplots of 10) and lower RMSPE in 28 of the 30.

## 4.6 Discussion

In this chapter I have extended the capabilities of full posterior inference for deep Gaussian processes through the incorporation of the Vecchia approximation. DGPs offer more flexibility than shallow GP surrogates as they are able to accommodate non-stationarity through the warpings of latent Gaussian layers. But inference is slow owing to cubic scaling of matrix decompositions. With a Vecchia approximation at each Gaussian layer, my fully-Bayesian

DGP enjoys computational costs scaling linearly-in-$n$. I demonstrated the superior UQ and predictive power of Vecchia-DGPs over both approximate DGP competitors and stationary GPs.

I envision many opportunities for extension. Most notably, I restricted my simulation studies to those where the noise parameter ($g$) was fixed to a small value. While many computer simulations are deterministic, others are increasingly stochastic in nature [3] and prompt estimation of this hyperparameter, and possibly others. My `deepgp` software is capable of estimating $g$ through additional Metropolis steps. In the Vecchia context, my implementation incorporates the $g$ parameter directly into the kernel by adding it to the diagonal of $\Sigma(x_i)$ and $\Sigma(X_{c(i)})$ in Eq. (4.2). Thus $g$ is subsumed into $B_i(W)$ and $\sigma_i^2(W)$ which in turn populate $U_w$ (4.4). As a demonstration of this capability, I provide a noisy simulated example in Section 4.4. Some authors separate responses into "true/latent" and "actual/observed" variables and caution against incorporating the noise parameter directly into the kernel [e.g., 22, 65], but I have not seen any drawbacks to my simplified approach in the DGP context. Furthermore, it may be possible to extend my DGP model to accommodate heteroskedastic noise through additional latent Gaussian random variables [10], although I caution that a model that is too flexible may fall prey to a signal–noise identifiability trap. One remedy is to provide for replicates in the design [11].

I restricted my empirical studies to a conditioning set size of $m = 25$, after finding limited advantage to larger $m$. This echoes other GP-Vecchia works which have found success with small conditioning sets [e.g., 22, 67, 122, 129]. Similarly, I entertained only two-layer DGPs. My experience with three-layer DGPs – mainly involving surrogate modeling – is that one of the latent layers settles into a near identity mapping, resulting in a lot more computation for no additional gain. In some cases, the added flexibility of a three-layer DGP can lead to overfitting and adversely affect predictions/UQ [Section 3.1.3]. My test functions and

real-data computer simulations may not be non-stationary enough to warrant two or more levels of latent warping.

Perhaps the most intriguing extension lies in the choice of the Vecchia ordering and conditioning sets. Common practice, as I embraced, involves simply fixing an ordering and conditioning, sometimes informed by the data as when NNs are scaled or warped. Many works have investigated the effects of different ordering/conditioning structures [e.g., 47, 65, 121], yet these analyses have all been *post hoc*. If one views the ordering and conditioning structure as components of the stochastic (data generating) process, there may be potential to *learn* these, either through maximization or similar MCMC sampling. In my simulation studies, updating the NN conditioning sets based on learned latent warpings did not affect posterior predictions, perhaps suggesting that inference for these would have similar null-effects. Yet I suspect this is because the learned latent warpings I have encountered tend to rotate and stretch inputs, resulting in minimal effects on the proximity of observations. I am continuing to "hunt" for examples where more flexible neighborhoods are valuable, since it seems like they should be.

# Chapter 5

# Extensions and Future Work

I provided a novel Bayesian framework for deep Gaussian process (DGP) surrogates which prioritizes uncertainty quantification (UQ). I demonstrated superior performance (both in predictive accuracy and UQ) over stationary GPs and various non-stationary competitors (including alternative inferential apparati for DGPs) on a variety of synthetic examples and real computer experiments. When computer simulations are computationally expensive, I provided novel methodology for sequential designs in three realms: variance reduction, Bayesian optimization, and contour location. When data is more abundant, I integrated the Vecchia approximation for faster computation without sacrificing DGP surrogate fidelity. Perhaps most importantly, I furnished an R package for easy implementation of all methods [110]. I also provided a public git repository with a plethora of reproducible examples, including all of those presented in this work.[1]

I embraced a fully-Bayesian scheme for DGPs by conducting MCMC sampling of the latent Gaussian layers and the kernel hyperparameters. In hindsight, the latent layers are the most important unkown variables, and the kernel hyperparameters have much smaller influence over the model. Full posterior integration of these hyperparameters may be overkill, and maximum-likelihood based alternatives like Expectation Maximization schemes may provide the same result with less computation [89].

DGPs excel when dynamics are non-stationary, but if the response surface is stationary

[1] https://bitbucket.org/gramacylab/deepgp-ex/

then a DGP surrogate requires a lot of extra computation for no additional gain. I relied on expert knowlege of the real-world computer simulations I entertained to posit whether non-stationary dynamics would be present. It would be beneficial to have methodology that could indicate whether non-stationary is present with minimal computation. Perhaps some diagnostic plots of a quick stationary GP surrogate would suffice [6].

The DGP surrogate I developed addresses non-stationarity in the response surface (i.e. the mean). It may be possible to upgrade the DGP to also account for non-stationarity in the noise level as done by Binois et al. [10] for stationary GPs. I caution against introducing too much flexibility in the surrogate model, as it could fall prey to signal-to-noise trade-offs. But adequate replication may circumvent this pitfall [11]. Most of the simulations I worked with were deterministic or low-noise, but there exist computer simulations with significant amounts of stochastic noise [3].

Although I explored three realms of sequential design, it would be beneficial to additionally explore sequential designs for calibration with DGPs. DGPs have been shown to perform well in calibration contexts [81]. Sequential design for calibration with stationary GPs has just recently come on the scene [72]. The combination of these two may prove very effective.

# Bibliography


[1] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[2] Shan Ba and V Roshan Joseph. Composite gaussian process models for emulating expensive functions. *The Annals of Applied Statistics*, pages 1838–1860, 2012.

[3] Evan Baker, Pierre Barbillon, Arindam Fadikar, Robert B Gramacy, Radu Herbei, David Higdon, Jiangeng Huang, Leah R Johnson, Pulong Ma, Anirban Mondal, et al. Analyzing stochastic computer models: A review with opportunities. *Statistical Science*, 37(1):64–89, 2022.

[4] Sudipto Banerjee, Alan E. Gelfand, Andrew O. Finley, and Huiyan Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 70(4):825–848, 2008. ISSN 13697412. doi: 10.1111/j.1467-9868.2008.00663.x.

[5] Stephen Barnett. *Matrix methods for engineers and scientists*. McGraw-Hill, 1979.

[6] Leonardo S Bastos and Anthony O'hagan. Diagnostics for gaussian process emulators. *Technometrics*, 51(4):425–438, 2009.

[7] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2021. URL https://CRAN.R-project.org/package=Matrix. R package version 1.3-4.

[8] James O Berger, Victor De Oliveira, and Bruno Sansó. Objective bayesian analysis

of spatially correlated data. *Journal of the American Statistical Association*, 96(456): 1361–1374, 2001.

[9] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[10] Mickael Binois, Robert B Gramacy, and Mike Ludkovski. Practical heteroscedastic gaussian process modeling for large simulation experiments. *Journal of Computational and Graphical Statistics*, 27(4):808–821, 2018.

[11] Mickaël Binois, Jiangeng Huang, Robert B. Gramacy, and Mike Ludkovski. Replication or exploration? sequential design for stochastic simulation experiments. *Technometrics*, 61(1):7–23, 2019. doi: 10.1080/00401706.2018.1469433.

[12] Luke Bornn, Gavin Shaddick, and James V Zidek. Modeling nonstationary processes through dimension expansion. *Journal of the American Statistical Association*, 107 (497):281–289, 2012.

[13] Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep gaussian processes for regression using approximate expectation propagation. In *International conference on machine learning*, pages 1472–1481. PMLR, 2016.

[14] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266 – 298, 2010. doi: 10.1214/09-AOAS285.

[15] David Cohn. Neural network exploration using optimal experiment design. In *Advances in Neural Information Processing Systems*, volume 6, pages 679–686. Morgan-Kaurmann, 1994.

[16] D Austin Cole, Ryan B Christianson, and Robert B Gramacy. Locally induced gaussian processes for large-scale simulation experiments. *Statistics and Computing*, 31(3):1–21, 2021.

[17] D Austin Cole, Robert B Gramacy, James E Warner, Geoffrey F Bomarito, Patrick E Leser, and William P Leser. Entropy-based adaptive design for contour finding and estimating reliability. *Journal of Quality Technology*, pages 1–18, 2022.

[18] Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep gaussian processes. In *International Conference on Machine Learning*, pages 884–893. PMLR, 2017.

[19] Kurt Cutajar, Mark Pullin, Andreas Damianou, Neil Lawrence, and Javier González. Deep gaussian processes for multi-fidelity modeling. *arXiv preprint arXiv:1903.07320*, 2019.

[20] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

[21] Abhirup Datta. Sparse cholesky matrices in spatial statistics. *arXiv preprint arXiv:2102.13299*, 2021.

[22] 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, 2016.

[23] Christophe Deissenberg, Sander Van Der Hoog, and Herbert Dawid. Eurace: A massively parallel agent-based model of the european economy. *Applied Mathematics and Computation*, 204(2):541–552, 2009.

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

[25] Vincent Dutordoir, Nicolas Knudde, Joachim van der Herten, Ivo Couckuyt, and Tom Dhaene. Deep gaussian process metamodeling of sequentially sampled non-stationary response surfaces. In *2017 Winter Simulation Conference (WSC)*, pages 1728–1739. IEEE, 2017.

[26] Vincent Dutordoir, Hugh Salimbeni, Eric Hambro, John McLeod, Felix Leibfried, Artem Artemev, Mark van der Wilk, James Hensman, Marc P Deisenroth, and ST John. Gpflux: A library for deep gaussian processes. *arXiv preprint arXiv:2104.05674*, 2021.

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

[28] Thomas D Economon, Francisco Palacios, Sean R Copeland, Trent W Lukaczyk, and Juan J Alonso. Su2: An open-source suite for multiphysics simulation and design. *Aiaa Journal*, 54(3):828–846, 2016.

[29] Dirk Eddelbuettel and Conrad Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71: 1054–1063, March 2014. URL http://dx.doi.org/10.1016/j.csda.2013.02.005.

[30] X Emery. The kriging update equations and their application to the selection of neighboring data. *Computational Geosciences*, 13(3):269–280, 2009.

[31] Arindam Fadikar, Dave Higdon, Jiangzhuo Chen, Bryan Lewis, Srinivasan Venkatramanan, and Madhav Marathe. Calibrating a stochastic, agent-based model using quantile-based emulation. *SIAM/ASA Journal on Uncertainty Quantification*, 6(4): 1685–1706, 2018.

[32] Jingjing Fei, Jing Zhao, Shiliang Sun, and Yan Liu. Active learning methods with deep gaussian processes. In *International Conference on Neural Information Processing*, pages 473–483. Springer, 2018.

[33] Andrew O Finley, Abhirup Datta, Bruce D Cook, Douglas C Morton, Hans E Andersen, and Sudipto Banerjee. Efficient algorithms for bayesian nearest neighbor gaussian processes. *Journal of Computational and Graphical Statistics*, pages 1–14, 2019. doi: 10.1080/10618600.2018.1537924.

[34] Andrew O Finley, Abhirup Datta, and Sudipto Banerjee. spnngp: R package for nearest neighbor gaussian process models. *Journal of Statistical Software*, 2020.

[35] AO Finley, H Sang, S Banerjee, and AE Gelfand. Improving the performance of predictive process modeling for large datasets. *Computational Statistics & Data Analysis*, 53(8):2873–2884, 2009.

[36] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

[37] Nathaniel Garton, Jarad Niemi, and Alicia Carriquiry. Knot selection in sparse gaussian processes with a variational objective function. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, pages 324–336, 2020.

[38] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[39] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[40] Robert B. Gramacy. tgp: An R package for bayesian nonstationary, semiparametric nonlinear regression and design by treed gaussian process models. *Journal of Statistical Software*, 19(9):1–46, 2007. URL http://www.jstatsoft.org/v19/i09/.

[41] Robert B. Gramacy. laGP: Large-scale spatial modeling via local approximate gaussian processes in R. *Journal of Statistical Software*, 72(1):1–46, 2016. doi: 10.18637/jss.v072.i01.

[42] Robert B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida, 2020. http://bobby.gramacy.com/surrogates/.

[43] Robert B Gramacy and Daniel W Apley. Local gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24 (2):561–578, 2015.

[44] Robert B Gramacy and Herbert K H Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

[45] Robert B. Gramacy and Herbert K. H. Lee. Adaptive design and analysis of supercomputer experiments. *Technometrics*, 51(2):130–145, 2009. doi: 10.1198/TECH.2009.0015.

[46] Robert B Gramacy, Annie Sauer, and Nathan Wycoff. Triangulation candidates for bayesian optimization. *Advances in Neural Information Processing Systems*, 35:35933–35945, 2022.

[47] Joseph Guinness. Permutation and grouping methods for sharpening gaussian process approximations. *Technometrics*, 60(4):415–429, 2018.

[48] Joseph Guinness. Gaussian process learning via fisher scoring of vecchia's approximation. *Statistics and Computing*, 31(3):1–8, 2021.

[49] Joseph Guinness, Matthias Katzfuss, and Youssef Fahmy. *GpGp: Fast Gaussian Process Computation Using Vecchia's Approximation*, 2021. URL https://CRAN.R-project.org/package=GpGp. R package version 0.4.0.

[50] Kai Habel, Raoul Grasman, Robert B. Gramacy, Pavlo Mozharovskyi, and David C. Sterratt. *geometry: Mesh Generation and Surface Tessellation*, 2022. URL https://CRAN.R-project.org/package=geometry. R package version 0.4.6.

[51] Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes. Inference in deep gaussian processes using stochastic gradient hamiltonian monte carlo. In *Advances in neural information processing systems*, pages 7506–7516, 2018.

[52] Matthew J Heaton, Abhirup Datta, Andrew O Finley, Reinhard Furrer, Joseph Guinness, Rajarshi Guhaniyogi, Florian Gerber, Robert B Gramacy, Dorit Hammerling, Matthias Katzfuss, et al. A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological and Environmental Statistics*, 24 (3):398–425, 2019.

[53] Ali Hebbal, Loic Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Nouredine Melab.

Bayesian optimization using deep gaussian processes with applications to aerospace system design. *Optimization and Engineering*, 22(1):321–361, 2021.

[54] D. M. Higdon, H. Lee, and C. Holloman. Markov chain Monte Carlo-based approaches for inference in computationally intensive inverse problems (with discussion). In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, editors, *Bayesian Statistics 7. Proceedings of the Seventh Valencia International Meeting*, pages 181–197. Oxford University Press, 2003.

[55] Dave Higdon, Jenise Swall, and John Kern. Non-stationary spatial modeling. *Bayesian statistics*, 6(1):761–768, 1999.

[56] Dave Higdon, Marc Kennedy, James C Cavendish, John A Cafeo, and Robert D Ryne. Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing*, 26(2):448–466, 2004.

[57] Muhammad F Izzaturrahman, Pramudita S Palar, Lavi Zuhal, and Koji Shimoyama. Modeling non-stationarity with deep gaussian processes: Applications in aerospace engineering. In *AIAA SCITECH 2022 Forum*, page 1096, 2022.

[58] L.R. Johnson. Microcolony and biofilm formation as a survival strategy for bacteria. *Journal of Theoretical Biology*, 251:24–34, 2008.

[59] Mark E Johnson, Leslie M Moore, and Donald Ylvisaker. Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26(2):131–148, 1990.

[60] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[61] V Roshan Joseph, Li Gu, Shan Ba, and William R Myers. Space-filling designs for robustness experiments. *Technometrics*, 61(1):24–37, 2019.

[62] Myeongjong Kang and Matthias Katzfuss. Correlation-based sparse inverse cholesky factorization for fast gaussian-process inference. *arXiv preprint arXiv:2112.14591*, 2021.

[63] Robert E Kass, Bradley P Carlin, Andrew Gelman, and Radford M Neal. Markov chain monte carlo in practice: a roundtable discussion. *The American Statistician*, 52 (2):93–100, 1998.

[64] Matthias Katzfuss. Bayesian nonstationary spatial modeling for very large datasets. *Environmetrics*, 24(3):189–200, 2013.

[65] Matthias Katzfuss and Joseph Guinness. A general framework for vecchia approximations of gaussian processes. *Statistical Science*, 36(1):124–141, 2021.

[66] Matthias Katzfuss, Joseph Guinness, Wenlong Gong, and Daniel Zilber. Vecchia approximations of gaussian-process predictions. *Journal of Agricultural, Biological and Environmental Statistics*, 25(3):383–414, 2020.

[67] Matthias Katzfuss, Joseph Guinness, and Earl Lawrence. Scaled vecchia approximation for fast computer-model emulation. *arXiv preprint arXiv:2005.00386*, 2020.

[68] Matthias Katzfuss, Marcin Jurek, Daniel Zilber, and Wenlong Gong. *GPvecchia: Scalable Gaussian-Process Approximations*, 2020. URL https://CRAN.R-project.org/package=GPvecchia. R package version 0.1.3.

[69] CG Kaufman, D Bingham, S Habib, K Heitmann, and JA Frieman. Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. *The Annals of Applied Statistics*, 5(4):2470–2492, 2011.

[70] Marc C Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3): 425–464, 2001.

[71] Hajime Kita, Kazuhisa Taniguchi, and Yoshihiro Nakajima. *Realistic Simulation of Financial Markets: Analyzing Market Behaviors by the Third Mode of Science*, volume 4. Springer, 2016.

[72] Scott Koermer, Justin Loda, Aaron Noble, and Robert B Gramacy. Active learning for simulator calibration. *arXiv preprint arXiv:2301.10228*, 2023.

[73] Issam H Laradji, Mark Schmidt, Vladimir Pavlovic, and Minyoung Kim. Efficient deep gaussian process models for variable-sized inputs. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019.

[74] Miguel Lázaro-Gredilla, Joaquin Quinonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[75] Luyao Lin, Derek Bingham, Floor Broekgaarden, and Ilya Mandel. Uncertainty quantification of a computer model for binary black hole formation. *The Annals of Applied Statistics*, 15(4):1604–1627, 2021.

[76] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[77] 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, 2020.

[78] Kailong Liu, Yi Li, Xiaosong Hu, Mattin Lucu, and Widanalage Dhammika Widanage. Gaussian process regression with automatic relevance determination kernel for calendar aging prediction of lithium-ion batteries. *IEEE Transactions on Industrial Informatics*, 16(6):3767–3777, 2020. doi: 10.1109/TII.2019.2941747.

[79] Xiaoyu Liu and Serge Guillas. Dimension reduction for gaussian process emulation: An application to the influence of bathymetry on tsunami heights. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):787–812, 2017.

[80] David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

[81] Sébastien Marmin and Maurizio Filippone. Deep Gaussian Processes for Calibration of Computer Models. *Bayesian Analysis*, pages 1 – 30, 2022. doi: 10.1214/21-BA1293. URL https://doi.org/10.1214/21-BA1293.

[82] Alexandre Marques, Remi Lam, and Karen Willcox. Contour location via entropy reduction leveraging multiple information sources. *Advances in neural information processing systems*, 31, 2018.

[83] Amandine Marrel, Bertrand Iooss, Beatrice Laurent, and Olivier Roustant. Calculations of sobol indices for the gaussian process metamodel. *Reliability Engineering & System Safety*, 94(3):742–751, 2009.

[84] David Marten, Jan Wendler, Georgios Pechlivanoglou, Christian Navid Nayeri, and Christian Oliver Paschereit. Qblade: an open source tool for design and simulation of horizontal and vertical axis wind turbines. *International Journal of Emerging Technology and Advanced Engineering*, 3(3):264–269, 2013.

[85] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of

three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[86] Piyush M Mehta, Andrew Walker, Earl Lawrence, Richard Linares, David Higdon, and Josef Koller. Modeling satellite drag coefficients with response surfaces. *Advances in Space Research*, 54(8):1590–1607, 2014.

[87] Arman Melkumyan and Fabio Tozeto Ramos. A sparse covariance function for exact gaussian process inference in large datasets. In *Twenty-first international joint conference on artificial intelligence*, 2009.

[88] Microsoft and Steve Weston. `foreach`*: Provides Foreach Looping Construct*, 2020. URL https://CRAN.R-project.org/package=foreach. R package version 1.5.0.

[89] Deyu Ming, Daniel Williamson, and Serge Guillas. Deep gaussian process emulation using stochastic imputation. *arXiv preprint arXiv:2107.01590*, 2021.

[90] Max D Morris and Toby J Mitchell. Exploratory designs for computational experiments. *Journal of statistical planning and inference*, 43(3):381–402, 1995.

[91] Iain Murray, Ryan Prescott Adams, and David J. C. MacKay. Elliptical slice sampling. In *The Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, volume 9 of *JMLR: W&CP*, pages 541–548. PMLR, 2010.

[92] Radford M Neal. Mcmc using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

[93] Jeremy Oakley. Estimating percentiles of uncertain computer code outputs. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 53(1):83–93, 2004.

[94] Jeremy E Oakley and Anthony O'Hagan. Probabilistic sensitivity analysis of complex models: a bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):751–769, 2004.

[95] Dean S Oliver, Luciane B Cunha, and Albert C Reynolds. Markov chain monte carlo methods for conditioning a permeability field to pressure data. *Mathematical Geology*, 29(1):61–91, 1997.

[96] Christopher J. Paciorek and Mark J. Schervish. Nonstationary covariance functions for gaussian process regression. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 273–280, Cambridge, MA, USA, 2003. MIT Press.

[97] Bandu Pamadi, Peter Covell, Paul Tartabini, and Kelly Murphy. Aerodynamic characteristics and glide-back performance of langley glide-back booster. In *22nd Applied Aerodynamics Conference and Exhibit*, page 5382, 2004.

[98] V Picheny, RB Gramacy, S Wild, and S Le Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In *Advances in Neural Information Processing Systems*, pages 1435–1443, 2016.

[99] Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[100] Majdi I Radaideh and Tomasz Kozlowski. Surrogate modeling of advanced computer simulations using deep gaussian processes. *Reliability Engineering & System Safety*, 195:106731, 2020.

[101] Dushhyanth Rajaram, Tejas G Puranik, S Ashwin Renganathan, WoongJe Sung,

Olivia Pinon Fischer, Dimitri N Mavris, and Arun Ramamurthy. Empirical assessment of deep gaussian process surrogate models for engineering problems. *Journal of Aircraft*, 58(1):182–196, 2021.

[102] Carl E Rasmussen and Zoubin Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in neural information processing systems*, volume 2, pages 881–888. MIT, 2002.

[103] Carl Edward Rasmussen. The infinite gaussian mixture model. In *In Advances in Neural Information Processing Systems 12*, pages 554–560. MIT Press, 2000.

[104] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Mass., 2005. ISBN 9780262182539.

[105] Patrick Roocks. Computing pareto frontiers and database preferences with the rpref package. *The R Journal*, 8(2):393–404, dec 2016. doi: 10.32614/RJ-2016-054. URL https://doi.org/10.32614/RJ-2016-054.

[106] Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. *arXiv preprint arXiv:1705.08933*, 2017.

[107] Andrea Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer physics communications*, 145(2):280–297, 2002.

[108] Paul D Sampson and Peter Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417): 108–119, 1992.

[109] TJ Santner, BJ Williams, and W Notz. *The Design and Analysis of Computer Experiments, Second Edition*. Springer–Verlag, New York, NY, 2018.

[110] Annie Sauer. `deepgp`: *Deep Gaussian Processes using MCMC*, 2020. URL https://CRAN.R-project.org/package=deepgp. R package version 1.1.0.

[111] Annie Sauer, Andrew Cooper, and Robert B Gramacy. Vecchia-approximated deep gaussian processes for computer experiments. *Journal of Computational and Graphical Statistics*, pages 1–14, 2022.

[112] Annie Sauer, Robert B Gramacy, and David Higdon. Active learning for deep gaussian process surrogates. *Technometrics*, 0(0):1–15, 2022. doi: 10.1080/00401706.2021.2008505.

[113] Alexandra M Schmidt and Anthony O'Hagan. Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758, 2003.

[114] Dino Sejdinovic, Heiko Strathmann, Maria Lomeli Garcia, Christophe Andrieu, and Arthur Gretton. Kernel adaptive metropolis-hastings. In *International conference on machine learning*, pages 1665–1673. PMLR, 2014.

[115] Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: active data selection and test point rejection. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, volume 3, pages 241–246. IEEE, 2000.

[116] Michael C Shewry and Henry P Wynn. Maximum entropy sampling. *Journal of applied statistics*, 14(2):165–170, 1987.

[117] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. *Advances in Neural Information Processing Systems 18*, pages 1257–1264, 2006. ISSN 1049-5258. URL http://www.gatsby.ucl.ac.uk/{~}snelson/SPGP{_}up.pdf.

[118] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959. Curran Associates Inc., 2012.

[119] Bret Stanford, Annie Sauer, Kevin Jacobson, and James Warner. Gradient-enhanced reliability analysis of transonic aeroelastic flutter. In *AIAA SCITECH 2022 Forum*, page 0632, 2022.

[120] Michael L. Stein. *Interpolation of spatial data*. Springer-Verlag, 1999. ISBN 0-387-98629-4.

[121] 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, 2004.

[122] Jonathan R Stroud, Michael L Stein, and Shaun Lysen. Bayesian and maximum likelihood estimation for gaussian processes on an incomplete lattice. *Journal of computational and Graphical Statistics*, 26(1):108–120, 2017.

[123] Furong Sun, Robert B Gramacy, Benjamin Haaland, Earl Lawrence, and Andrew Walker. Emulating satellite drag from large simulation experiments. *SIAM/ASA Journal on Uncertainty Quantification*, 7(2):720–759, 2019.

[124] S Surjanovic and D Bingham. Virtual library of simulation experiments: test functions and datasets. http://www.sfu.ca/~ssurjano, 2013.

[125] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12): 2295–2329, 2017.

[126] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[127] John Vassberg, Mark Dehaan, Melissa Rivers, and Richard Wahls. Development of a common research model for applied cfd validation studies. In *26th AIAA applied aerodynamics conference*, page 6919, 2008.

[128] Aldo V Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):297–312, 1988.

[129] Luhuan Wu, Geoff Pleiss, and John Cunningham. Variational nearest neighbor gaussian processes. *arXiv preprint arXiv:2202.01694*, 2022.

[130] Nathan Wycoff, Mickaël Binois, and Robert B Gramacy. Sensitivity prewarping for local surrogate modeling. *arXiv preprint arXiv:2101.06296*, 2021.

[131] Jie Yang and Diego Klabjan. Bayesian active learning for choice models with deep gaussian processes. *IEEE Transactions on Intelligent Transportation Systems*, 22(2): 1080–1092, 2020.

[132] Boya Zhang, D Austin Cole, and Robert B Gramacy. Distance-distributed design for gaussian process surrogates. *Technometrics*, 63(1):40–52, 2021.

[133] Yichi Zhang, He Zhao, Irene Hassinger, L Catherine Brinson, Linda S Schadler, and Wei Chen. Microstructure reconstruction and structural equation modeling for computational design of nanodielectrics. *Integrating Materials and Manufacturing Innovation*, 4(1):14, 2015.

# Appendices

# Appendix A

# Derivations

## A.1 Partitioned Matrix Inverse (V1)

Following the properties of partitioned matrices [e.g., 5], with $\mathbf{k}_n(w_{n+1}) = K_{\theta_y}(W_n, w_{n+1})$,

$$C_{n+1} = \begin{bmatrix} C_n & \mathbf{k}_n(w_{n+1}) \\ \mathbf{k}_n^\top(w_{n+1}) & 1+g \end{bmatrix} \quad \text{yields} \quad C_{n+1}^{-1} = \begin{bmatrix} \left[C_n^{-1} + \mathbf{h}\mathbf{h}^\top v\right] & \mathbf{h} \\ \mathbf{h}^\top & v^{-1} \end{bmatrix}$$

where $v = 1 + g - \mathbf{k}_n^\top(w_{n+1}) C_n^{-1} \mathbf{k}_n(w_{n+1})$ and $\mathbf{h} = -v^{-1} C_n^{-1} \mathbf{k}_n(w_{n+1})$.

## A.2 IMSE Derivation

Here I extend the closed-form IMSE derivation of Binois et al. [11] and Joseph et al. [61] to allow integration under uniform measure over the general domain $W_i \in [a_i, b_i]$ for $i = 1, \ldots, p$ for the model $Y \sim N_n(\mathbf{0}, K_{\theta_y}(W) + g\mathbb{I})$ and isotropic Gaussian kernel (1.2). Application to MCMC samples requires indexing with iteration $(t)$, but I drop this notation here for convenience. I set $\Sigma(w) = 1$ instead of $\Sigma(w) = 1 + g$ to target the variance of the mean.

Using $W_{n+1}$ and $C_{n+1}$ as described in Section 3.1.1, and additionally denoting $\mathbf{k}_{n+1}(w) =$

$K_{\theta_y}(W_{n+1}, w)$, IMSE may be expressed as follows.

$$\begin{aligned}
\text{IMSE}(W_{n+1}) &= \int_{a_p}^{b_p} \cdots \int_{a_1}^{b_1} \Sigma_Y(w)\ dw_1 \ldots dw_p \\
&= \prod_{i=1}^{p} (b_i - a_i) \mathbb{E}\left[\Sigma_Y(w)\right] \\
&= \prod_{i=1}^{p} (b_i - a_i) \mathbb{E}\left[\hat{\tau}^2 \left(1 - \mathbf{k}_{n+1}^\top(w) C_{n+1}^{-1} \mathbf{k}_{n+1}(w)\right)\right] \\
&= \hat{\tau}^2 \prod_{i=1}^{p} (b_i - a_i) \left[1 - \mathbb{E}\left[\mathbf{k}_{n+1}^\top(w) C_{n+1}^{-1} \mathbf{k}_{n+1}(w)\right]\right]
\end{aligned}$$

This expectation reduces to a trace of matrix products following Lemma 3.1 of Binois et al. [11],

$$\mathbb{E}\left[\mathbf{k}_{n+1}^\top(w) C_{n+1}^{-1} \mathbf{k}_{n+1}(w)\right] = \text{tr}\left(C_{n+1}^{-1} \mathbf{H}\right),$$

where the elements of $\mathbf{H}$ are defined as

$$\begin{aligned}
\mathbf{H}_{jk} &= \prod_{i=1}^{p} \int_{a_i}^{b_i} K_{\theta_y}(w_{j,i}, w) K_{\theta_y}(w_{k,i}, w)\ dw_i \\
&= \prod_{i=1}^{p} \int_{a_i}^{b_i} \exp\left(\frac{-(w_{j,i} - w)^2}{\theta_y}\right) \exp\left(\frac{-(w_{k,i} - w)^2}{\theta_y}\right)\ dw_i \\
&= \prod_{i=1}^{p} \int_{a_i}^{b_i} \exp\left(-\frac{2}{\theta_y}\left(w - \frac{w_{j,i} + w_{k,i}}{2}\right)^2 - \frac{1}{2\theta_y}(w_{j,i} - w_{k,i})^2\right)\ dw_i \\
&= \prod_{i=1}^{p} \left[\exp\left(-\frac{(w_{j,i} - w_{k,i})^2}{2\theta_y}\right) \int_{a_i}^{b_i} \exp\left(-\frac{2}{\theta_y}\left(w - \frac{w_{j,i} + w_{k,i}}{2}\right)^2\right)\ dw_i\right] \\
&= \prod_{i=1}^{p} \left[\sqrt{\frac{\pi\theta_y}{2}} \exp\left(-\frac{(w_{j,i} - w_{k,i})^2}{2\theta_y}\right) \int_{a_i}^{b_i} \mathcal{N}\left(w \mid \mu = \frac{w_{j,i} + w_{k,i}}{2}, \sigma^2 = \frac{\theta_y}{4}\right)\ dw_i\right] \\
&= \left(\frac{\pi\theta_y}{2}\right)^{\frac{p}{2}} \prod_{i=1}^{p} \left\{\exp\left(-\frac{(w_{j,i} - w_{k,i})^2}{2\theta_y}\right) \left[\Phi\left(\frac{2b_i - w_{j,i} - w_{k,i}}{\sqrt{\theta_y}}\right) - \Phi\left(\frac{2a_i - w_{j,i} - w_{k,i}}{\sqrt{\theta_y}}\right)\right]\right\}.
\end{aligned}$$

Above, $w_{j,i}$ is the $j^{th}$ element of the $i^{th}$ node of $W$ and $\Phi$ is the standard Gaussian cumulative distribution function (CDF).

## A.3 ALC Derivation

Here, I detail the re-expression of ALC from Eq. (3.3) in terms of the new latent element, $w_{n+1}$. Again, I drop $(t)$ indexing to streamline. Quantities $\mathbf{k}_n(w)$, $v$, and $\mathbf{h}$ are defined in Appendix A.1 and yield partitioned representation of $\mathbf{k}_{n+1}^\top(w) = \left[\mathbf{k}_n^\top(w) \mid z\right]$ where $z = K_{\theta_y}(w_{n+1}, w)$. Tedious matrix multiplication gives

$$
\begin{aligned}
\mathbf{k}_{n+1}^\top(w) C_{n+1}^{-1} \mathbf{k}_{n+1}(w) &= \left[\mathbf{k}_n^\top(w) \mid z\right] \begin{bmatrix} C_n^{-1} + \mathbf{h}\mathbf{h}^\top v & \mathbf{h} \\ \mathbf{h}^\top & v^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{k}_n(w) \\ z \end{bmatrix} \\
&= \left[\mathbf{k}_n^\top(w) \left(C_n^{-1} + \mathbf{h}\mathbf{h}^\top v\right) + z\mathbf{h}^\top \mid \mathbf{k}_n^\top(w)\mathbf{h} + zv^{-1}\right] \begin{bmatrix} \mathbf{k}_n(w) \\ z \end{bmatrix} \\
&= \mathbf{k}_n^\top(w)(C_n^{-1} + \mathbf{h}\mathbf{h}^\top v)\mathbf{k}_n(w) + z\mathbf{h}^\top \mathbf{k}_n(w) + \mathbf{k}_n^\top(w)\mathbf{h}z + v^{-1}z^2 \\
&= \mathbf{k}_n^\top(w) C_n^{-1} \mathbf{k}_n(w) + v(\mathbf{k}_n^\top(w)\mathbf{h})^2 + 2z\mathbf{k}_n^\top(w)\mathbf{h} + v^{-1}z^2 \\
&\propto v(\mathbf{k}_n^\top(w)\mathbf{h})^2 + 2z\mathbf{k}_n^\top(w)\mathbf{h} + v^{-1}z^2,
\end{aligned}
$$

producing the following ALC

$$
\mathrm{ALC}(w_{n+1} \mid \mathcal{W}_{\mathrm{ref}}) \propto \sum_{w \in \mathcal{W}_{\mathrm{ref}}} \hat{\tau}^2 \left[v(\mathbf{h}^\top \mathbf{k}_n(w))^2 + 2z\mathbf{h}^\top \mathbf{k}_n(w) + v^{-1}z^2\right].
$$

## A.4 Partitioned Matrix Inverse (V2)

The inverse of a partitioned matrix follows [5]:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}^{-1} \quad \text{where} \quad \begin{aligned} A_{11} &= \left(B_{11} - B_{12}B_{22}^{-1}B_{21}\right)^{-1} \\ A_{12} &= -B_{11}^{-1}B_{12}\left(B_{22} - B_{21}B_{11}^{-1}B_{12}\right)^{-1} \\ A_{21} &= -B_{22}^{-1}B_{21}\left(B_{11} - B_{12}B_{22}^{-1}B_{21}\right)^{-1} \\ A_{22} &= \left(B_{22} - B_{21}B_{11}^{-1}B_{12}\right)^{-1} \end{aligned}$$

## A.5 Vecchia Posterior Predictive Moments

I aim to predict $\mathcal{Y}$ at locations $\mathcal{W}$ conditioned on observed $Y$ and $W$. I assume a zero-mean Gaussian process prior distribution,

$$\begin{bmatrix} Y \\ \mathcal{Y} \end{bmatrix} \sim \mathcal{N}_{n+n_p}\left(0, \ \Sigma_{\text{stack}}\right) \quad \text{where} \quad \Sigma_{\text{stack}} = \Sigma\left(\begin{bmatrix} W \\ \mathcal{W} \end{bmatrix}\right) = \begin{bmatrix} \Sigma(W) & \Sigma(W,\mathcal{W}) \\ \Sigma(\mathcal{W},W) & \Sigma(\mathcal{W}) \end{bmatrix}.$$

Under the Vecchia-approximation, I decompose the precision matrix using the sparse upper-lower Cholesky decomposition, with entries populated according to Eq. (4.4),

$$U_{\text{stack}} = \begin{bmatrix} U_w & U_{w,\mathcal{W}} \\ 0 & U_{\mathcal{W}} \end{bmatrix} \quad \text{such that} \quad \Sigma_{\text{stack}} = \left(U_{\text{stack}}U_{\text{stack}}^\top\right)^{-1} = \left(\begin{bmatrix} U_wU_w^\top + U_{w,\mathcal{W}}U_{w,\mathcal{W}}^\top & U_{w,\mathcal{W}}U_{\mathcal{W}}^\top \\ U_{\mathcal{W}}U_{w,\mathcal{W}}^\top & U_{\mathcal{W}}U_{\mathcal{W}}^\top \end{bmatrix}\right)^{-1}.$$

I aim to find a closed-form solution to the posterior predictive moments (1.4),

$$\mathcal{Y} \mid Y, W \sim \mathcal{N}_{n_p}\left(\mu^\star, \Sigma^\star\right) \quad \text{for} \quad \begin{aligned} \mu^\star &= \Sigma(\mathcal{W},W)\Sigma(W)^{-1}Y \\ \Sigma^\star &= \Sigma(\mathcal{W}) - \Sigma(\mathcal{W},W)\Sigma(W)^{-1}\Sigma(W,\mathcal{W}), \end{aligned}$$

which can avoid dense covariance matrices by instead relying on elements of the sparse $U_{\text{stack}}$. The simplification of $\Sigma^\star$ follows directly from the partitioned matrix inverse (Appendix A.4),

$$U_{\mathcal{W}}U_{\mathcal{W}}^\top = \left(\Sigma(\mathcal{W}) - \Sigma(\mathcal{W}, W)\Sigma(W)^{-1}\Sigma(W, \mathcal{W})\right)^{-1} \quad \Longrightarrow \quad \Sigma^\star = \left(U_{\mathcal{W}}U_{\mathcal{W}}^\top\right)^{-1}.$$

The calculation of $\mu^\star$ first involves simplification of $\Sigma(W)$ and $\Sigma(\mathcal{W}, W)$, again using partitioned matrix inverses (Appendix A.4)

$$\begin{aligned}
\Sigma(W) &= \left(U_wU_w^\top + U_{w,\mathcal{W}}U_{w,\mathcal{W}}^\top - U_{w,\mathcal{W}}U_{\mathcal{W}}^\top\left(U_{\mathcal{W}}U_{\mathcal{W}}^\top\right)^{-1}U_{\mathcal{W}}U_{w,\mathcal{W}}^\top\right)^{-1} \\
&= \left(U_wU_w^\top\right)^{-1} \\
\Sigma(\mathcal{W}, W) &= -\left(U_{\mathcal{W}}U_{\mathcal{W}}^\top\right)^{-1}U_{\mathcal{W}}U_{w,\mathcal{W}}^\top\left(U_wU_w^\top + U_{w,\mathcal{W}}U_{w,\mathcal{W}}^\top - U_{w,\mathcal{W}}U_{\mathcal{W}}^\top\left(U_{\mathcal{W}}U_{\mathcal{W}}^\top\right)^{-1}U_{\mathcal{W}}U_{w,\mathcal{W}}^\top\right)^{-1} \\
&= -\left(U_{\mathcal{W}}U_{\mathcal{W}}^\top\right)^{-1}U_{\mathcal{W}}U_{w,\mathcal{W}}^\top\left(U_wU_w^\top\right)^{-1}
\end{aligned}$$

Together, these yield

$$\begin{aligned}
\mu^\star &= \Sigma(\mathcal{W}, W)\Sigma(W)^{-1}Y \\
&= -\left(U_{\mathcal{W}}U_{\mathcal{W}}^\top\right)^{-1}U_{\mathcal{W}}U_{w,\mathcal{W}}^\top\left(U_wU_w^\top\right)^{-1}U_wU_w^\top Y \\
&= -\left(U_{\mathcal{W}}^\top\right)^{-1}U_{w,\mathcal{W}}^\top Y.
\end{aligned}$$