Efficient Computer Experiment Designs for Gaussian Process Surrogates

D. Austin Cole

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 M. Higdon

Xinwei Deng

Leanna L. House

May 24, 2021

Blacksburg, Virginia

Keywords: inducing points, active learning, big data, kriging, reliability

Efficient Computer Experiment Designs for Gaussian Process Surrogates

D. Austin Cole

(ABSTRACT)

Due to advancements in supercomputing and algorithms for finite element analysis, today's computer simulation models often contain complex calculations that can result in a wealth of knowledge. Gaussian processes (GPs) are highly desirable models for computer experiments for their predictive accuracy and uncertainty quantification. This dissertation addresses GP modeling when data abounds as well as GP adaptive design when simulator expense severely limits the amount of collected data. For data-rich problems, I introduce a localized sparse covariance GP that preserves the flexibility and predictive accuracy of a GP's predictive surface while saving computational time. This locally induced Gaussian process (LIGP) incorporates latent design points, *inducing points*, with a local Gaussian process built from a subset of the data. Various methods are introduced for the design of the inducing points. LIGP is then extended to adapt to stochastic data with replicates, estimating noise while relying upon the unique design locations for computation. I also address the goal of identifying a contour when data collection resources are limited through entropy-based adaptive design. Unlike existing methods, the entropy-based contour locator (ECL) adaptive design promotes exploration in the design space, performing well in higher dimensions and when the contour corresponds to a high/low quantile. ECL adaptive design can join with importance sampling for the purpose of reducing uncertainty in reliability estimation.

Efficient Computer Experiment Designs for Gaussian Process Surrogates

D. Austin Cole

(GENERAL AUDIENCE ABSTRACT)

Due to advancements in supercomputing and physics-based algorithms, today's computer simulation models often contain complex calculations that can produce larger amounts of data than through physical experiments. Computer experiments conducted with simulation models are sought-after ways to gather knowledge about physical problems but come with design and modeling challenges. Examples include predicting a satellite's trajectory in real time and predicting the impact damage of a spacesuit. In this dissertation, I address both data size extremes – building prediction models with large data sets and designing computer experiments when scarce resources limit the amount of data. For the former, I introduce a strategy of constructing a series of models including small subsets of observed data (with inputs and their responses) along with a set of inputs without observed responses (*inducing points*). This methodology also contains the ability to perform calculations with only unique data locations when replicates exist in the data. The locally induced model produces accurate predictions while saving computing time. Various methods are introduced to decide the locations of these inducing points. The focus then shifts to designing an experiment for the purpose of accurate prediction around a contour, a region of input values where the response is close to a specific quantity. A experimental design approach is detailed that selects new sample locations one-at-a-time through a function to maximize the amount of information gain in the contour region for the overall model. This work is combined with an existing method to estimate the true volume of the contour.

Dedication

to Stacie

Acknowledgments

During my journey to receive a Ph.D., there have been many intelligent people who supported my studies. I would not be the statistician I am today without their encouragement and support. I first want to thank my adivisor, Robert B. Gramacy, for his supervision and mentoring during my research. Dr. Gramacy's patience, wisdom, and passion for computer experiments have been key to my growth as a researcher. I also extend my gratitude to my other committee members, David. M Higdon, Xinwei Deng, and Leanna, L. House. I am grateful for their encouragement and valuable questions and suggestions regarding my research.

During my graduate studies, I have also had several mentors and collaborators who have shaped me. I want to thank Jennifer Van Mullekom for mentoring me during my years serving in the Statistical Application and Innovations Group (SAIG). With Dr. Van Mullekom's support, I was afforded a variety of collaborative and teaching opportunities with SAIG that helped refine my communication skills. I also extend my gratitude to James E. Warner, Geoffrey F. Bomarito, Patrick E. Leser, and William D. Leser at NASA Langley Research Center. I owe much of my refined coding skills to the research I have completed with their help.

Thanks you's are due to my fellow classmates that have walked alongside me for my research journey. I am thankful for Jiangeng Huang, Furong Sun, Boya Zhang, and Adam Edwards as the original members of Dr. Gramacy's lab when I joined the group. They each displayed patience and insight, teaching me the basics of Gaussian processes and debugging code, as well as a open mind to bounce around ideas with. I also thank Nathan Wycoff, Ryan Christianson, and Annie Sauer, current members of the Gramacy lab, for their support and

feedback after each lab presentation.

I am tremendously grateful for the financial support of the National Science Foundation and the Engineering Research & Analysis program at NASA for their funding to support my research.

I would be remiss to conclude without thanks my family for their support during these challenging years. I am tremendously grateful for the support that my parents and sister gave me throughout this entire journey, especially when first making the transition from teaching to a career in statistics. I also thank my wife, Stacie, for her love, patience, and sacrifice during my marriage as I worked to complete this degree.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Curiosity and the desire for technological advancement are parts of human nature. Early examples span from simple machines like the wheel or pulley to architectural feats of the Egyptian pyramids and Machu Picchu. In more recent human history, great minds like Galileo Galilei and Louis Pasteur turned to scientific experimentation to answer lofty questions and achieve great technological improvements in physics and chemistry. Yet it was not until the first half of the twentieth century that Ronald A. Fisher proposed and conducted *statistically-based experiments*, paving the way for a new era for scientific discovery. Fisher's work on agriculture experiments serves as the foundation for modern statistical experimentation, modeling, and analysis. He recognized the issue of variability in observed data and the importance of separating signal from noise. Many techniques for designing experiments based on the type of modeling and analysis (e.g. factorial designs, response surface methodology) are still used by modern statisticians.

Statistically-based experimentation has expanded far beyond just agricultural applications. Today, statistics is sought after to determine cause-and-effect relationships in health care, business, and manufacturing. Each field presents its own set of experimental challenges, but a common hindrance is having the resources necessary to conduct physical experiments. In the modern computer age, one such solution is conducting computer experiments.

1.1.1 Computer Experiments

After decades of applying statistical, mathematical, and other scientific methods, contemporary scientists and engineers have deep understandings about complex processes of interest today. Many scientists and engineers leverage that knowledge through building sophisticated computer simulations to act as a viable substitute for a physical process. The fields of chemistry, public health, finance, and aerospace use computer simulations to model protein reactions to chemical compounds, the spread of a disease within a social group, stock trading strategies, and airplane drag. Computer experimentation provides structure to gather data for analysis in such applications.

Scientists and engineers conduct computer experiments by running a simulator with a set of inputs and observing the output. These simulations, which often incorporate physics-based or other complex models, allow their users to expand the bounds of experimentation. For example, simulations may provide opportunities to gather data about a drug's risk to pregnant women without physical trials or the accuracy of a rocket's projectile without risking thousands of lives. The development of high performance computing allow computer experiments to be run across multiple machines, which increases the time savings compared to conducting physical experiments. In addition, the sheer nature of forgoing the physical elements and manpower needed for a physical experiment provides a significant reduction of financial resources.

1.1.2 Modern Motivating Examples

Many challenging examples of computer simulations come from engineering and biological applications. The descriptions that follow serve as such examples, including common challenges of a high-dimensional set of inputs and/or heteroskedastic output.

Robot arm

The SARCOS data set (Rasmussen and Williams, 2006; Vijayakumar and Schaal, 2000) is based on an inverse dynamics problem of an actual anthropomorphic robot arm (rather than a computer simulation). The training and testing sets of roughly 45,000 and 4,500 samples, respectively, is commonly used in the machine learning community. It includes 21 inputs, seven joint positions, velocities, and accelerations. The output corresponds to the torque command used with the robot. The relationship in the data is highly non-linear and the set of training inputs is not space-filling, instead lying on a lower-dimensional manifold.

Satellite drag

The work of tracking satellite trajectories and collision avoidance is a key concern of institutions and companies that rely on satellites for communication and security. Many researchers also rely on the positioning of satellites for experimentation and data collection. Modeling satellite drag is a key component of the success of each of the aforementioned interests. Researchers at Los Alamos National Lab developed the *Test Particle Monte Carlo* simulator to model the low Earth orbit environment of a satellite for specific atmospheric elemental compositions (atomic oxygen, molecular oxygen, atomic nitrogen, molecular nitrogen, helium, and hydrogen). The data sets described in Sun et al. (2019a), Mehta et al. (2014), Gramacy (2020, Chapter 2.3.3) and the Git repo `https://bitbucket.org/gramacylab/tpm/src` contain sets of 1 million simulated runs of the Hubble Space Telescope for each of the seven elements. The simulator contains 8 inputs: velocity, surface temperature, atmospheric temperature, yaw, pitch, normal energy AC, tangential momentum AC, and panel angle. The simulator produces the coefficient of drag for the satellite.

Susceptible-infected-recovery (SIR) epidemics managements

SIR models are commonly used for cost-benefit analysis of public health measures to combat the spread of communicable diseases such as influenza or Ebola. Models can be very complex, including many inputs for various health measures. One of the simplest versions of said model includes two inputs: the initial number susceptible individuals and the number of infected individuals. The model's response is the expected aggregate number of infected-days across Markov chain trajectories until the epidemic ends. The output is stochastic, with input-dependent-noise. For implementation of this model, I rely on the function `sirEval` in the `R` package `hetGP` (Binois and Gramacy, 2018).

Ocean oxygen concentration

Another stochastic simulator models oxygen concentration in the ocean's depths (McKeague et al., 2005). This highly heteroskedastic simulator approximates the solution of an advection-diffusion equation from computational fluid dynamics with Monte Carlo methods. The four inputs are latitude and longitude coordinates and two diffusion coefficients. The output is the ocean oxygen concentration.

Spacesuit impact damage

As a part of the Artemis mission to send astronauts to the moon and Mars, the National Aeronautics and Space Administration (NASA) is developing the next generation spacesuit, known as the Exploration Extravehicular Mobility Unit (xEMU). A large focus on the redesigned spacesuit is increasing the astronaut's mobility and lessening the weight and burden of wearing the suit for extended periods of time. During this design phase, engineers need to assess the impact safety of the suit, accounting for uncertainty in the composition and

thickness of materials in hard upper torso (HUT) the suit. In addition, various impact loads (due to projectiles, falls, etc.) must be considered for a holistic understanding of the suit's reliability. Due to the common restrictions of supplies and time, NASA researchers use a computer simulation to model the impact damage on the HUT instead of conducting many destructive physical tests.

The `Z-2` spacesuit assembly model, developed at the University of Delaware for a previous spacesuit generation, serves as a worthy substitute for experimentation while the final simulator for xEMU is being finalized. The `Z-2` simulator uses the LS-DYNA finite element method software (Gladman et al., 2007) to simulate the impact caused by a fall. The MAT162 material model (Gama et al., 2009; Haque, 2017) estimates progressive damage to the composite materials in the suit with LS-DYNA. The simulation model includes a set of material parameters that define the onset of damage in the spacesuit as well as three input parameters related to the velocity and damage limits of the projectile. Through calibration and sensitivity analysis (Warner et al., 2021), three variables of interest stand out, while the other four are fixed in later experiments. These parameters are coefficients of strain softening for fiber damage (in two directions) and matrix crack and delamination. The impact velocity of the fall is also considered as a key parameter in the impact damage modeling. The output from the `Z-2`/MAT162 model provides a time lapsed set of force measurements for each set of inputs. From the set of functional responses, the main quantity of interest is the maximum force.

Chapter 2

Review of Literature

2.1 Gaussian process modeling

The history of modeling random functions with stochastic processes extends from the area of spacial statistics in the second half of the twentieth century (Cressie, 2015; Matheron, 1963; Stein, 2012). Kriging, as the spatial community calls it, is known in the statistics community as Gaussian stochastic process modeling or Gaussian process regression. Using a Gaussian process prior over a random function equates to assuming that any finite set of observations follows a multivariate normal (MVN) distribution. Gaussian processes (GPs) provide a flexible, nonparametric regression method that is widely used for computer experiments (Forrester et al., 2008; Santner et al., 2018). The strength of GPs lies in their ability to produce accurate predictions through interpolation accompanied with uncertainty quantification.

2.1.1 Gaussian process overview

To get more technical, a Gaussian stochastic process places a prior of $z(\mathbf{x}) \sim GP(\mu(\mathbf{x}), \Sigma(\mathbf{x}, \mathbf{x}^{'}))$ on the random function $z(\mathbf{x})$, with $\mathbf{x} \in \mathbb{R}^d$. Here the prior is defined solely by the first two

moments of a Normal distribution for any pair of inputs $\mathbf{x}$ and $\mathbf{x}'$:

$$\mu(\mathbf{x}) = \mathbb{E}[z(\mathbf{x})], \qquad \Sigma(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(z(\mathbf{x}) - \mu(\mathbf{x}))(z(\mathbf{x}') - \mu(\mathbf{x}'))^\top] \tag{2.1}$$

The true power of a GP is through its ability to produce accurate predictions, conditioned on observed data. For N observations included in $D_N = (\mathbf{X}_N, \mathbf{Y}_N)$, the posterior predictive (or conditioned multivariate-normal) distribution for a set of inputs $\mathbf{x}'$ is $y(\mathbf{x}'|D_N) \sim \mathcal{N}(\mu_N(\mathbf{x}'), \Sigma_N(\mathbf{x}'))$. Assuming $\mu(\mathbf{X}_N) = 0$ and $\mu(\mathbf{x}') = 0$, a common simplification, the predictive equations are:

$$\begin{aligned} \mu_N(\mathbf{x}'|D_N) &= \Sigma(\mathbf{x}', \mathbf{X}_N)\Sigma(\mathbf{X}_N, \mathbf{X}_N)^{-1}\mathbf{Y}_N \\ \sigma_N^2(\mathbf{x}'|D_N) &= \Sigma(\mathbf{x}', \mathbf{x}') - \Sigma(\mathbf{x}', \mathbf{X}_N)\Sigma(\mathbf{X}_N, \mathbf{X}_N)^{-1}\Sigma(\mathbf{x}', \mathbf{X}_N)^\top \end{aligned} \tag{2.2}$$

In the spatial statistics arena, the equations in (2.2) are referred to as the *kriging equations* (Cressie, 2015). These predictions also hold the property of being the best linear unbiased predictors (BLUP) from the frequentist perspective (Santner et al., 2018).

2.1.2 Covariance functions

By assuming a zero-mean GP, more focus and value is placed on the covariance matrix $\Sigma(\mathbf{x}, \mathbf{x}')$. For starters, any well-defined covariance matrix Σ must be positive definite, meaning

$$\mathbf{x}^\top \Sigma(\cdot, \cdot)\mathbf{x} > 0 \qquad \forall \mathbf{x} \in \mathbb{R}^d \tag{2.3}$$

The covariance matrix is filled out element-wise, where the i, j entry of $\Sigma(\mathbf{X}, \mathbf{X}')$ is defined as the product of the covariance or kernel function $k(\mathbf{x}_i, \mathbf{x}'_j)$ and a scale (or variance) parameter

τ^2:

$$\Sigma(\mathbf{X}, \mathbf{X}') = \tau^2 \mathbf{K}(\mathbf{X}, \mathbf{X}') \qquad \mathbf{K}^{ij} = k(\mathbf{x}_i, \mathbf{x}'_j). \tag{2.4}$$

Here $\mathbf{x}_i$ is a vector of the ith row of matrix $\mathbf{X}$ and $\mathbf{x}'_j$ is a vector of the jth row of matrix $\mathbf{X}'$. When it comes to defining a GP the specification of the kernel function is critical.

The fundamental idea behind the construction of a covariance matrix through the definition of a kernel function $k(\cdot, \cdot)$ is that data are highly correlated when they lie close to each other in the input space. This accompanies a general assumption of the GP that the underlying relationship in the data is smooth. A *stationary* GP contains the assumption that data and the random process continues to exhibit the same correlation relationship throughout the entire input space. In other words, the kernel function $k(\mathbf{x}, \mathbf{x}') \equiv k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')$ when $||\mathbf{x} - \mathbf{x}'|| = ||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}'||$. In this way the value of the kernel function is defined entirely by the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$, $d(\mathbf{x}, \mathbf{x}')$, rather than the actual input locations $\mathbf{x}$ and $\mathbf{x}'$. the Euclidean distance between sets of points.

An *isotropic* kernel function goes one step further, assuming the relationship between $\mathbf{x}$ and $\mathbf{x}'$ is invariant to translations or rotations. This simplifies the kernel function to be determined by a single one-dimensional distance measure between $\mathbf{x}$ and $\mathbf{x}'$, also called a radial basis function. A common kernel function is based on the kernel of a Gaussian distribution, and is referred to as the Gaussian or squared-exponential kernel:

$$k_\theta(\mathbf{x}, \mathbf{x}') = \exp\left\{ -\frac{||\mathbf{x} - \mathbf{x}'||^2}{\theta} \right\}. \tag{2.5}$$

The kernel function $k_\theta(\mathbf{x}, \mathbf{x}')$ in equation (2.5) includes the hyperparameter θ known as the lengthscale, used to adjust the rate of the distanced-based correlation. Supposing isotropy in the data is a strong assumption that can greatly affect the quality of a GP fit. Separable kernels provide more flexibility by defining the kernel function by a series of distance mea-

sures, one for each input dimension. In its separable form, the squared-exponential kernel function uses a vectorized lengthscale $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$:

$$k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') = \exp\left\{ -\sum_{k=1}^{d} \frac{(x_k - x_k')^2}{\theta_k} \right\}. \tag{2.6}$$

Use of the squared-exponential kernel produces a smooth correlation structure that is infinitely differentiable (Stein, 2012). The Matérn function (Stein, 2012) is another kernel function that allows tuning of the smoothness and differentiability through the value ρ,

$$k_\rho(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\rho}}{\Gamma(\rho)} \left(||\mathbf{x} - \mathbf{x}'|| \sqrt{\frac{2\rho}{\theta}} \right)^{\rho} K_\rho \left(||\mathbf{x} - \mathbf{x}'|| \sqrt{\frac{2\rho}{\theta}} \right). \tag{2.7}$$

Here Γ and K_ρ are the Gamma and modified Bessel functions respectively. Common choices for ρ are $\{3/2, 5/2\}$, which are once and twice differentiable. As $\tau^2 \to \infty$, $k_\rho(\mathbf{x}, \mathbf{x}') = \exp\left\{ \frac{||\mathbf{x}-\mathbf{x}'||^2}{2\theta} \right\}$, becoming a member of the Gaussian family. As with the squared-exponential kernel, a separable version of the Matérn function also exists.

The family of compactly supported kernels seeks to create sparsity in the covariance matrix, simplifying matrix calculations. This is done by defining $k^{(d_{\max})}(\mathbf{x}, \mathbf{x}') = 0$ when $d(\mathbf{x}, \mathbf{x}') > d_{\max}$. Kernels can also be constructed by the product of multiple kernel functions. This can be applied component-wise as well. This definition allows for the flexibility of any other kernel function when $d(\mathbf{x}, \mathbf{x}') \leq d_{\max}$. It is important to note that $d_{\max}$ must also be tuned or determined by the user.

Many other kernel functions exist, incorporating other forms like the power-exponential or addressing issues like non-stationarity (Abrahamsen, 1997; Genton, 2001; Higdon et al., 1999; Rasmussen and Williams, 2006; Stein, 2012). The selection of a particular kernel function and its hyperparameters is a key component of fitting a GP not to be overlooked.

Any choice should be based on prior belief of the function space, including impressions of smoothness, decay of the correlation in relation to the input distances, and the signal-to-noise ratio.

2.1.3 Modeling with noisy observations

In many examples, even when gathering data from computer simulations, data consists of noisy observations, deviating from the true random mean function. To address this in a GP, a *nugget* η is often added to the diagonal of the correlation matrix

$$\Sigma(\mathbf{x}, \mathbf{x}') = \tau^2 \left(\mathbf{K}(\mathbf{x}, \mathbf{x}') + \eta \delta_{\mathbf{x},\mathbf{x}'}\right) \tag{2.8}$$

where τ^2 is the scale parameter, $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ is the kernel matrix, and $\delta_{\mathbf{x},\mathbf{x}'}$ is the kronecker delta. This nugget represents the observational error, adding random noise into the GP.

Adding a nugget along the diagonal of the correlation matrix acts as a smoother, helping break the GP from its usual strict interpolation of data points. The use of a nugget acts in a similar way to a *jitter* (Neal, 1998), often used when the data is deterministic. The use of a nugget helps with the matrix decomposition through eigenvalues (Neal, 1997). It also provides further stability in the matrix important for the surrogate (Gramacy and Lee, 2012).

2.2 Gaussian process approximations for large data sets

Thanks to improvements in data gathering and computational resources, today's data sets are often orders of magnitude larger than those commonly analyzed in the twentieth century.

Possessing a rich data set to analyze can be a statistician's dream, provided their analysis tools can handle the size of the data. GP are not immune to this concern. In order for GP prediction using equation (2.2), the $N \times N$ covariance matrix $\Sigma(X_N, X_N)$ must be inverted. Without special structure in the covariance matrix, this inversion has a cubic cost, $\mathcal{O}(N^3)$. This limits contemporary computing resources to invert matrices with the number of rows only in the thousands (Rasmussen and Williams, 2006, Chapter 8).

Researchers from machine learning, geostatistics, and computer experiments communities are working to reduce the computational burden of GPs. One strategy is to induce sparsity in the covariance matrix. Wilson and Nickisch (2015) leverage Kronecker and Toeplitz methods to structure the covariance based on a number of latent design points (inducing points). They then use structured kernel interpolation to approximate the covariance between the inducing points and data locations, creating a sparse covariance structure. Other methods use the Lanczos approximation to construct a low-rank covariance matrix (Dong et al., 2017; Gardner et al., 2018b; Pleiss et al., 2018). Kaufman et al. (2011) show that compactly supported kernels (CSKs) can be used to combine basis-expanded mean structures with reduced rank covariances, but this leads to only modest computational enhancements. Further work by Genton (2001) suggests that reordering of the terms in a CSK-based covariance matrix closer to the diagonal enhances the computational savings.

Sparsity in the precision matrix is another strategy for reducing the computational burden of GP modeling. Much of this work is inspired by the Vecchia approximation (Vecchia, 1988). Datta et al. (2016) introduced Nearest-Neighbor Gaussian processes (NNGP), which use a conditional distribution framework and nearest neighbors as a reference set for their NNGP prior. In related work, Katzfuss and Guinness (2021) extend the Vecchia approximation of the Cholesky factor of the precision matrix by conditioning on both latent and observed variables. They use directed acyclic graph models to represent the sparsity

of their approximation. Further advancements extend the Vecchia approximation to high dimensional data (Katzfuss et al., 2020).

A different angle used to handle large data sets is a "divide-and-conquer" approach. In the spatial statistics community, Kim et al. (2005) sought to address how GPs could still be used when there are sharp changes in the covariance function, not adhering to the traditional smooth and stationary assumptions. Their piece-wise GPs used Voronoi tessellations to perform a fully Bayesian partition of the design space. They apply their work in the two-dimensional space, creating disjoint partitions that come together to form an overall piece-wise stationary GP. Unfortunately, constructing such partitions is computationally expensive.

Gramacy and Lee (2008) developed Bayesian treed Gaussian processes (TGP) to also partition the data, but in a recursive sense that is aligned with the axes. This allows greater interpretability of the partitions and the potential to expand far beyond two dimensions. In addition, TGP can incorporate parallelized multi-core computing, saving wall-clock time. It also induces statistical independence in the partitions, allowing for regime changes or other non-stationary behavior.

2.2.1 Inducing point methods

Imposing a low-rank structure on the covariance matrix is one direct approach to speedy GP approximation in the face of big N. Wahba (1990, Chapter 7) and Poggio and Girosi (1990) first proposed selecting local data subsets for splines, referred to as *centers*. Later the concept was applied to GPs, which used likelihood approximations to select a subset of *basis functions* for a covariance matrix (Smola and Bartlett, 2001). Csató and Opper (2002) introduced *basis vectors* to induce sparsity in the covariance matrix, which also works for

discontinuous likelihoods. A simpler approach developed by Seeger et al. (2003) created a sparse GP regression that uses an *active set*, or subset of the data. The active set serves as reference locations to construct a low-rank covariance matrix.

Snelson and Ghahramani (2006) coined the term *psuedo inputs* when proposing that these reference locations not be restricted to a subset of the data. Quiñonero and Rasmussen (2005) and Rasmussen and Williams (2006, Chapter 8) provide unifying perspectives for sparse approximate GPs, with the former referring to these latent reference variables as *inducing inputs*. In the spatial community, Banerjee et al. (2008) applied similar techniques to develop *predictive processes*. In this dissertation, I use the term *inducing points* to refer to the latent design points related to all of these methods.

The popularity of inducing points has grown considerably since the mid-2000s. Wilson and Nickisch (2015) developed KISS-GP, which combined inducing points with structured kernel interpolation and Kroecker and Toeplitz methods, allowing the number of inducing points M to be greater than N. Further work by Pleiss et al. (2018) extended KISS-GP to include Lanczos approximation to the $M \times M$ matrix that does not depend on predictive locations. Blitzkriging uses Kronecker structure to calculate the precision matrix by requiring that variational covariance and kernels are products in each dimension (Nickson et al., 2015). Applications of inducing points also include GPs with multidimensional change surfaces (Herlands et al., 2016) and deep GPs (Damianou and Lawrence, 2013).

Inducing point notation and equations

Let $\mathbf{\Psi}_M = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_M)^\top$ be M inducing points in $\mathcal{X}$, the same design space as $\mathbf{X}_N$, but they are not restricted to the elements of $\mathbf{X}_N$. Denote $\mathbf{K}_M$ as a kernel matrix built from $\mathbf{\Psi}_M$ and $k(\cdot, \cdot)$; similarly, write $\mathbf{k}_{NM}$ as cross evaluations of the kernel between $\mathbf{X}_N$ and $\mathbf{\Psi}_M$.

Inducing point methods (Seeger et al., 2003; Snelson and Ghahramani, 2006) commonly base GP approximations on the so-called Nyström approximation (Williams and Seeger, 2001): $\mathbf{K}_N \approx \mathbf{K}_N^{(M)} = \mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top$. This uses $M \ll N$ references in $\mathbf{\Psi}_M$ to induce similar structure $\mathbf{K}_N^{(M)}$ to the calculated covariance between each pair of design points.

Not all $\mathbf{\Psi}_M$ are equal. There are trivial pathological choices, but space-filling designs work well. Snelson and Ghahramani (2006) suggested that $\mathbf{\Psi}_M$ could be tuned through derivative-based optimization of the log-likelihood. They also introduced a diagonal correction on the Nyström approximation

$$\mathbf{\Sigma}_N^{(M)} = \tau^2(\mathbf{K}_N^{(M)} + \mathbf{U}psilon_N) = \tau^2\left(\mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top + \mathrm{Diag}\{\mathbf{K}_N - \mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top + \mathbf{U}psilon_N\}\right), \tag{2.9}$$

where $\mathbf{U}psilon_N$ is a diagonal matrix of the nugget (η) and/or jitter (ϵ). This ensures that $\mathbf{K}_N$ and $\mathbf{K}_N^{(M)}$ contain the same diagonal entries so when $\mathbf{\Psi}_M \equiv \mathbf{X}_N$, $\mathbf{\Sigma}_N^{(M)}$ in equation (2.9) reduces to the standard GP covariance (2.4). Both approximations allow decomposition of $\mathbf{\Sigma}_N^{(M)}$ through Woodbury matrix identities (Harville, 1998), producing:

$$\begin{aligned} \mathbf{\Sigma}_N^{-1(M)} &= \tau^{-2}\left(\Omega_N^{-1(M)} - \Omega_N^{-1(M)}\mathbf{k}_{NM}\mathbf{Q}_M^{-1(N)}\mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\right) \\ \log|\mathbf{\Sigma}_N^{(M)}| &= \log(\tau^2) + \log|\mathbf{K}_M + \mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\mathbf{k}_{NM}| - \log|\mathbf{K}_M| + \mathbf{1}_N^\top\log(\Omega_N^{(M)})\mathbf{1}_N, \end{aligned} \tag{2.10}$$

where $\mathbf{1}_N$ is a vector of N ones. Above, $\mathbf{Q}_M^{(N)} = \mathbf{K}_M + \mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\mathbf{k}_{NM}$ and $\Omega_N^{(M)} = \mathrm{Diag}\{\mathbf{K}_N - \mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top + \mathbf{U}psilon_N\}$. Inferring hyperparameters is achieved by maxi-

mizing the logarithm of the MVN likelihood $Y_N \sim \mathcal{N}\left(\mathbf{0}, \tau^2(\Omega_N^{(M)} + \mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top)\right)$:

$$\begin{aligned}\ell(\mathbf{X}, \mathbf{Y}, \mathbf{\Psi}_M, \tau^2, \theta, \eta) &= -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_N| - \frac{1}{2}\mathbf{Y}_N^\top \mathbf{\Sigma}_N^{-1}\mathbf{Y}_N \\ &= \text{const.} + N\log(\tau^2) + \log|\mathbf{K}_M + \mathbf{k}_{NM}^\top \Omega_N^{-1(M)}\mathbf{k}_{NM}| - \log|\mathbf{K}_M| + \log|\Omega_N^{(M)}| \\ &\quad + \tau^{-2}\mathbf{Y}_N^\top\left(\Omega_N^{-1(M)} - \Omega_N^{-1(M)}\mathbf{k}_{NM}\mathbf{Q}_M^{-1(N)}\mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\right)\mathbf{Y}_N.\end{aligned} \tag{2.11}$$

Early work (Seeger et al., 2003; Snelson and Ghahramani, 2006) suggest maximizing the log-likelihood to jointly optimize the locations of the inducing points, while Garton et al. (2020) suggests sequentially selecting the inducing points with the marginal likelihood with respect to the covariance parameters. Finley et al. (2009) use spatially averaged predictive variance. Titsias (2009) instead uses the Kullback-Leibler divergence between the inducing point model (with a variational distribution) and the posterior distribution to find inducing point locations. Further discussion of the challenges related to inducing point optimization is included in Chapter 3.

Following equation (2.9), the posterior predictive equations for the sparse approximate GP are Gaussian with

$$\begin{aligned}&\text{mean} && \mu_{M,N}(\mathbf{x}') = \mathbf{k}_M^\top(\mathbf{x}')\mathbf{Q}_M^{-1(N)}\mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\mathbf{Y}_N \\ &\text{and scale} && \sigma_{M,N}^2(\mathbf{x}') = \tau^2\left(k(\mathbf{x}', \mathbf{x}') + \upsilon - \mathbf{k}_M^\top(\mathbf{x}')(\mathbf{K}_M^{-1} - \mathbf{Q}_M^{-1(N)})\mathbf{k}_M(\mathbf{x}')\right),\end{aligned} \tag{2.12}$$

where $\mathbf{k}_M(\mathbf{x}') = k_\theta(\mathbf{\Psi}_M, \mathbf{x}')$ and υ is the nugget and/or jitter.

2.2.2 Local approximate Gaussian processes

Instead of adapting the GP framework to handle the whole data set at once, a *local approximate GP* (LAGP; Gramacy and Apley, 2015) considers approximating GPs of separate local

data subsets based on each predictive location $\mathbf{x}'$. Recall that with typical inverse-distance based correlation such as equation (2.5) ensures training data inputs $\mathbf{X}_N$ far from each $\mathbf{x}'$ provide little added value to the underlying predictor. Thus, I can construct subsets that contain most of the valuable correlation that are much smaller than N. For example, suppose that $(\mathbf{X}_n(\mathbf{x}'), \mathbf{Y}_n(\mathbf{x}'))$ represents an n-sized subset, or *neighborhood*, of the training data nearby $\mathbf{x}'$ (e.g. comprised of nearest neighbors (NNs)). Given a reasonable hyperparameterization, prediction can follow the equations in (2.12) using $(\mathbf{X}_n(\mathbf{x}'), \mathbf{Y}_n(\mathbf{x}'))$ rather than the full $(\mathbf{X}_N, \mathbf{Y}_N)$. Even though the inverse calculations are still cubic, having $n \ll N$ can potentially provide drastic computational savings.

It is important to note that his framework relies upon determining the subset size n and neighborhood $\mathbf{X}_n(\mathbf{x}')$. Because floating point operations (flops) grow quickly with n, this value is usually restricted by computational limitations. A default in the `laGP` software is $n = 50$ (Gramacy, 2016). With a NN subdesign, using a fixed n as originally suggested by Emery (2009) in a 2d geostatistics setting, is sub-optimal by several criteria (Stein, 2012; Vecchia, 1988). However, exhaustively searching among all $\binom{N}{n}$ alternatives for each $\mathbf{x}'$ is combinatorially infeasible.

Greedy neighborhood selection via active learning Cohn (ALC) approximately minimizes mean-squared error (MSE), a criterion common in surrogate modeling settings (Gramacy and Apley, 2015) . Through this ALC optimization and update, care is taken to ensure computational demands do not exceed $\mathcal{O}(n^2)$ so that the entire scheme's flops are not worse in order than using NNs (i.e., cubic in n). A template design for the neighborhood, could cut down the overall cost of building neighborhoods, without exhaustive search of $\mathbf{X}_N \setminus \mathbf{X}_n(\mathbf{x}')$. Such a design, involving a simple shifting/scaling for each $\mathbf{x}'$ does not currently exist, in part due to the non-uniform nature of global designs $\mathbf{X}_N$.

Selecting the size of the local design is directly linked to the topology of the data. More

wiggly test problems benefit from more reactive dynamics offered by smaller n. But in less wiggly setting, n much larger than the default of $n = 50$ can be a deal-breaker on speed grounds regardless of accuracy boosts. A brief exploration of local design size is discussed in Section 3.4.3.

Since its introduction, LAGP has become a benchmark competitor to new works in the area of scalable GPs due to its flexibility, accurate predictions, and time savings (Edwards and Gramacy, 2020; Katzfuss et al., 2020). LAGP has been directly applied to a wide variety of large-scale data problems, including calibration in radiative shock dynamics (Gramacy et al., 2015), emulation of satellite drag (Sun et al., 2019a), and spatial-temporal modeling for solar irradiance (Sun et al., 2019b).

2.3 Modeling and Design of computer experiments

Computer experiments seek to explore physical systems through a computational and mathematical model $\mathcal{T}$. This model maps a set of inputs $\{\mathbf{X}, \mathbf{U}\}$ and outputs $\mathbf{Y}^{(\mathcal{T})}$,

$$\mathbf{Y}^{(\mathcal{T})} = \mathcal{T}(\mathbf{X}, \mathbf{U}), \tag{2.13}$$

to mirror the real phenomena, event, or system. The input-output relationship emulated by computer models is usually complex and based in scientific understanding and powerful computing abilities. Often a computer model contains two sets of inputs, both of which can be high-dimensional. The set $\mathbf{X}$ is the series of inputs corresponding to the physical characteristics of the process which are measureable and controllable. The remaining input set $\mathbf{U}$ is known as the tuning or calibration variables that are not directly observable from a physical experiment. The act of efficiently selecting the best values for the input variables $\mathbf{X}$

and building accurate predictors with a reliable estimation of uncertainty is key in computer experimentation (Fang et al., 2005; Forrester et al., 2008; Santner et al., 2018).

Special attention is often needed to understand the uncertainty around the input set $\mathbf{U}$. *Bayesian calibration* seeks to quantify all sources of uncertainty in order to enhance the statistical model's (i.e. GP's) ability to predict with accurate measures of uncertainty. The Kennedy and O'Hagan (KOH) Bayesian calibration framework is the standard method for such a goal (Kennedy and O'Hagan, 2001; Kennedy and O'Hagan, 2001). The spirit of the KOH calibration framework is to represent the true process $Y^{\mathcal{Z}}(\cdot)$, computer model $Y^{\mathcal{T}}(\cdot)$, computer model discrepancy $b(\mathbf{x})$, and the true value of the calibration parameter(s) $\mathbf{u}^*$ with

$$Y^{\mathcal{Z}}(\mathbf{x}) = Y^{\mathcal{T}}(\mathbf{x}, \mathbf{u}^*) + b(\mathbf{x}) + \epsilon, \tag{2.14}$$

where ϵ is the random noise in the field measurements. Such calibration is contingent upon having a combination of field data in addition to data generated from the computer model. Details on how to integrate both data sources for calibration of $\mathbf{U}$ is outlined in Higdon et al. (2004).

2.3.1 Deterministic computer experiments

Historically, computer experiments consisted of computationally-intensive runs of a computer model producing deterministic output given a series of inputs (Sacks et al., 1989a,b). The nature of receiving deterministic output is a distinguishing characteristic of computer experiments compared to the noisy observations of physical experiments. Traditional statistical techniques of replication, randomization, and linear regression modeling did not seem as necessary. Instead, non-linear regression methods and interpolators like GPs are often sought for modeling and prediction.

Model-independent designs

Some experimental designs incorporate little knowledge about the model. *Model-independent designs* do not assume much, outside of the model being flexible enough to interpolate the design and data. A popular type of model-independent designs for computer experiments is Latin hypercube sampling (LHS) designs (Iman and Conover, 1980; McKay et al., 2000; Stein, 1987). An $N \times d$ LHS design of size N partitions each of the d input coordinates into N equally sized intervals and ensures that a single design point lies within each interval. LHS designs guarantee that the marginals are uniformly distributed. They are not unique and can pose problems when the number of dimensions is high.

Other space-filling designs focus on optimizing a distance function $d(\mathbf{x}, \mathbf{x}')$ to place design points to cover the design space $\mathcal{X}$. Maximin and minimax designs are distance-based designs that fill the design space by avoiding the worst distance settings (Johnson et al., 1990). Let $d(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{d} |x_k - x'_k|$ be the rectangular Euclidean distance and $\mathbf{D} \in [0,1]^d$ be the scaled unit design space. Minimax distanced designs seek to minimize the farthest pairwise distance between points

$$\arg\min_{\mathbf{D}} \max_{\mathbf{x} \in \mathcal{X}} \min_{i} d(\mathbf{x}, \mathbf{x}_i), \qquad i = 1, \ldots, N. \tag{2.15}$$

Maximin distance designs seek to do the converse, by maximizing the closest pairwise distance between points

$$\arg\max_{\mathbf{D}} \min_{i,j} d(\mathbf{x}_i, \mathbf{x}_j), \qquad i, j = 1, \ldots, N. \tag{2.16}$$

The maximin LHS design is a hybrid of the two space-filling methods that selects points within each interval according to the maximin criterion (Morris and Mitchell, 1995). Maxmin

LHS designs contain optimal space-filling properties and are thus among the most popular space-filling design choices. These designs can be generated quickly in the `lhs` (Carnell, 2019) and `DiceDesign` (Dupuy et al., 2015) packages in `R` (R Core Team, 2021).

New space-filling designs continue to be created to address new variations and challenges of computer experiments. Maximum projection designs extend the one-dimensional uniformity of LHS designs to larger subspaces (Joseph et al., 2015). Some computer model include both categorical and numerical predictors. New sliced space-filling designs (Qian and Wu, 2009) and sliced Latin hypercube designs (Qian, 2012) adapt the existing designs to include a combination of variable types. Maximin "sliced" LHD (SLHD) hybrids also provide an upgrade to include categorical variables (Ba et al., 2015).

Model-dependent designs

Another class of experimental designs, *model-dependent designs*, seeks to take advantage of the information obtained from existing statistical models built upon data. The model-dependent design philosophy of sequential design and response surface methodology, has a rich history in experimental design (Box and Draper, 1987, 2007; Myers et al., 2016).

When generating such designs, a popular goal is to minimize variance in the model. Mean squared prediction error (MSE) is defined as $\text{MSE}(\hat{y}(\mathbf{x})) = \mathbb{E}[\hat{y}(\mathbf{x}) - Y^Z(\mathbf{x})] \equiv \sigma_N^2(\mathbf{x})$ for a GP model. The integrated mean-squared prediction error (IMSE) seeks to select a design that will minimize the predictive variance across the design space (Sacks et al., 1989a,b). The IMSE criterion

$$\text{IMSE} = \mathbb{E}[\text{MSE}(\hat{y}(\mathbf{x}))] = \int_{\mathcal{X}} \frac{\sigma_N^2(\mathbf{x})}{\tau^2} w(\mathbf{x}) d\mathbf{x} \tag{2.17}$$

includes the flexibility to include prior information on the design space (from a Bayesian

perspective) or a weight through the function $w(\mathbf{x})$ (Sacks et al., 1989b). In Chapter 3, I derive a closed-form expression of the IMSE with $w(\mathbf{x})$ being a Gaussian kernel.

Another popular metric used to create a design is entropy, which is directly related to the amount of "information" contained in the data's design density. Maximizing entropy increases the uniformity of the design locations and the "surprise" of the response. In contrast, a low entropy (high information) would place the design points close to each other, resulting in little surprise in the observed responses. For a given probability density $p(\mathbf{x})$, entropy is defined as

$$H(X) = -\int_{\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}. \tag{2.18}$$

A solution to determine a maximum entropy design originated for spatial models (Shewry and Wynn, 1987) before being introduced for computer experiments (Currin et al., 1991; Mitchell and Scott, 1987).

2.3.2 Stochastic computer experiments

Despite the tradition of focusing on deterministic computer simulations, the need is growing for modeling strategies with noisy data. Today's computing advancements provide opportunities for stochastic computer models such as queueing systems and agent based models. More complicated dynamical systems and more areas of uncertainty contribute to the noise now observed in modern computer experiments. As the noise increases, the signal-to-noise ratio decreases, adding to the challenge of learning the signal. Replication is a classical and proven method to investigate the noise through the model's measure of pure error (Myers et al., 2016). Yet there is not a one-size-fits all effective and efficient strategy for replication.

Ankenman et al. (2010) laid the groundwork of separating signal from noise with replication through *stochastic kriging*. The authors use a moment-based approach that assumes

noise is present everywhere and changes drastically through the design space. From a design perspective, stochastic kriging provide a strategy for IMSE optimal design with replication. Stochastic kriging requires a substantial amount of replication throughout the design space to gather enough degrees of freedom for pure error estimation, which can be a costly requirement.

A variation of traditional GPs, heteroskedastic GPs (HetGP) provide a full likelihood-based inference framework that models data with input-dependent noise (Binois et al., 2018a). In HetGP, a latent GP is fit to the noise structure on the inputs. This provides the ability to extrapolate the noise structure where there is no replication. All of this makes HetGP more flexible to the amount of replication in the design. In addition, the GP structure takes advantage of replication and the well-known Woodbury identity to reduce the inference computational cost from the full scale N to the scale of unique $\bar{N}$ design locations, $\mathcal{O}(N^3)$ to $\mathcal{O}(\bar{N}^3)$.

Another strategy for modeling signal and noise is *quantile kriging* (Plumlee and Tuo, 2014). Unlike the previous methods, it does not assume a Gaussian or any other structure on the outputs. Instead it develops an emulative distribution on the outputs' quantiles. Plumlee and Tuo (2014) define an asymptotically optimal rate of convergence for their quantile kriging framework. Without the distributional assumption, quantile kriging is more flexible than other methods, but at the cost of more replication to obtain the quantiles of the full distribution for stochastic outputs.

2.3.3 Sequential learning for computer experiments

The statistical idea of sequential design has existed for decades, being a fundamental component of response surface methodology (Box and Draper, 1987; Myers et al., 2016). More

recently the machine learning community has developed great interest in *active learning* (i.e. sequential or adaptive design) and *Bayesian optimization* techniques. In both cases, the basis is around leveraging the current model's information (based on existing data) to greedily select the next sample/batch of inputs used to gather more data. To make such decisions, they rely on the optimization of an *acquisition function*.

Sequential design

Back in the 1990s, MacKay (1992) suggested maximizing predictive variance for sequential design related to neural networks. But it was not until Seo et al. (2000) that Active Learning McKay (ALM) was termed and applied to computer experiments. The application of sequentially selecting the $n+1^{st}$ point $\mathbf{x}_{n+1}$ based on the observed data $(\mathbf{X}_n, \mathbf{Y}_n)$ can be expressed as

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \sigma_n^2(\mathbf{x}), \tag{2.19}$$

where $\sigma_n^2(\mathbf{x})$ is the GP predictive variance (2.2). For a fairly straightforward criterion, using ALM for repeated sequential selections approximates a maximum entropy design (MacKay, 1992). This design is not perfect. Just because the variance is high does not guarantee more information will be learned from sampling in that region (like in non-constant noise problems). A more globally-focused criterion like IMSE is helpful, but a closed-form solution only exists when the design space $\mathcal{X}$ is rectangular.

Cohn (1994) was the first to suggest an acquisition function to measure the reduction in variance for a particular set of reference locations in the nonparametric regression context, focused again on neural networks. ALC was coined by Seo et al. (2000), becoming a sequential version of IMSE that approximates a full A-optimal design. The focus is on using what is modernly referred to as a lookahead scheme and taking advantage of the variance GP

predictive equation relying solely on the design points and not the responses (conditioned on the hyperparameters). ALC seeks to maximize the reduction in variance:

$$\Delta\sigma_n^2(\mathbf{x}_{n+1}) = \int_{\mathbf{x}\in\mathcal{X}} \sigma_n^2(\mathbf{x}) - \ddot{\sigma}_{n+1}^2(\mathbf{x})d\mathbf{x}. \tag{2.20}$$

Here $\ddot{\sigma}_{n+1}^2(\mathbf{x}) = \tau_n^2(1+g^{(n)} - k^\top(\mathbf{X}_{n+1}, \mathbf{x})\mathbf{K}_{n+1}^{-1}k(\mathbf{X}_{n+1}, \mathbf{x}))$, the lookahead predictive variance conditioned on the existing model's hyperparameters, and $\mathcal{X}$ is the entire (or a subset of the) design space. Maximizing (2.20) is equivalent to the following simplified acquisition function:

$$\mathbf{x}_{n+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} \int_{\mathbf{x}\in\mathcal{X}} \ddot{\sigma}_{n+1}^2(\mathbf{x})d\mathbf{x}. \tag{2.21}$$

In practice the integral in equation (2.21) is approximated with a sum, though a closed-form expression exists if the integral is over a rectangular design space. Both ALM and ALC follow similar tendencies in their sequential selection, but due to the more global nature of ALC, its global maxima is more often in the interior of the design space.

Other design methods focus on GP hyperparameter estimation. Maximization of the Fisher information of certain GP hyperparameters, in particular the lengthscale θ, is discussed in Gramacy and Apley (2015). The equations involve second derivatives and expectations, which can become rather cumbersome. But in general sequential design creates a strong feedback loop for hyperparameter estimation. Zhang et al. (2019) suggest using a betadist or random design (not a sequential design) which will provide a variety of pairwise distances between design points to help estimate the lengthscale.

Bayesian optimization

The area of Bayesian optimization has become popular recently (Ginsbourger and Le Riche, 2010; González et al., 2016; Gramacy et al., 2016; Lam et al., 2016; Letham and Bakshy, 2019; Letham et al., 2019; Picheny et al., 2016). Early ideas related to Bayesian optimization include Mockus et al. (1978) and Booker et al. (1999). The idea of optimization is a foundation of response surface methodology (Box and Draper, 1987, 2007), where the goal is usually to find the global maximum of a surface. However in Bayesian optimization, the goal is to identify the global minimum of a blackbox function $\mathcal{T}(\cdot)$, like a computer simulation model. Specifically, the desire is to find

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} \mathcal{T}(\mathbf{x}), \tag{2.22}$$

over a (often rectangular) design space $\mathcal{X}$. The assumption is there is no derivative information about $\mathcal{T}(\cdot)$–all the information about the function comes through the model evaluations. And often the function is expensive to evaluate, which is why Bayesian optimization strategies rely on sequential design.

A popular metric for Bayesian optimization of deterministic functions is expected improvement (EI), the basis of the efficient global optimization algorithm (Jones et al., 1998). Let $\mathcal{T}^n_{min} = \min\{y_1, \ldots, y_n\}$ be the lowest objective function of $\mathcal{T}^n(\cdot)$ observed so far. Then the potential improvement of sampling at the input location $\mathbf{x}$ is denoted by

$$I(\mathbf{x}) = \max\{0, \mathcal{T}^n_{min} - Y(\mathbf{x})\}, \tag{2.23}$$

where $Y(\mathbf{x})$ is the predictive distribution (2.2) conditioned on the observed data $(\mathbf{X}_n, \mathbf{Y}_n)$. $I(\mathbf{x})$ is a random variable that measures the amount a response $Y(\mathbf{x})$ *could* be below $\mathcal{T}^n_{min}$.

The expectation of equation (2.23) is equivalent to the following closed-form expression

$$\mathbb{E}[I(\mathbf{x})] = (\mathcal{T}^n_{min} - \mu_n(\mathbf{x}))\Phi\left(\frac{\mathcal{T}^n_{min} - \mu_n(\mathbf{x})}{\sigma_n(\mathbf{x})}\right) + \sigma_n(\mathbf{x})\phi\left(\frac{\mathcal{T}^n_{min} - \mu_n(\mathbf{x})}{\sigma_n(\mathbf{x})}\right), \tag{2.24}$$

where Φ and ϕ are the Gaussian cdf and pdf. Serendiptitouly, EI weighs exploitation close to the global minima with exploration of areas where the predictive variance is high.

The interest in this research area is growing. Other Bayesian optimization criteria have been developed, including the knowledge gradient (Wu et al., 2017) and integrated expected conditional improvement (Gramacy, 2011). Further work is focused on how to perform optimization with known or unknown constraints in the model (Lee et al., 2011; Picheny and Ginsbourger, 2014).

2.4 Reliability estimation using simulations

When modeling physical structures, there is often uncertainty in some of the simulator's parameters. This can take many forms: variation in material composition, instrument measurement error, or simply a distribution on a covariate. Coupled with this uncertainty in the input parameters is the common desire to assess the reliability of a structure, guarding against collapse, puncture, or failing of an electronic piece of equipment. Structural reliability describes the behavior of a structure under various load conditions of the parameters (Rajashekhar and Ellingwood, 1993).

Say that a simulator $\mathcal{T} : \mathbb{R}^d \to \mathbb{R}$ is such that $\mathcal{T}(\mathbf{x}) = y$, where $\mathbf{x}$ is a vector of d inputs with output y. Let $\mathbb{F}$ be a distribution over $\mathbf{x} \in \mathcal{X}$, where f is the density representing uncertainties in the inputs $\mathbf{x}$. The structure's reliability is defined through a *limit state function*, $g : \mathbb{R} \to \mathbb{R}$, where a failure in the structure is such that $g(y) > 0$ (Melchers and

Beck, 2018). The probability of failure (α) is defined as

$$\alpha = \mathbb{E}_X[g(\mathcal{T}(\mathbf{x})) > 0] = \int_{\mathbf{x}\in\mathcal{X}} \mathbb{1}_{g(\mathcal{T}(\mathbf{x}))>0} d\mathbb{F}, \tag{2.25}$$

where $\mathbb{1}$ is the indicator function.

Monte Carlo (MC) sampling is the traditional sampling technique to estimate $\hat{\alpha}_{\mathrm{MC}}$, forgoing the integration with a sum of $\acute{N}$ independently and identically distributed drawn simulation runs.

$$\hat{\alpha}_{\mathrm{MC}} = \frac{1}{\acute{N}} \sum_{i=1}^{\acute{N}} \mathbb{1}_{g(\mathcal{T}(x_i))>0} \tag{2.26}$$

While $\hat{\alpha}_{\mathrm{MC}}$ is an asymptotically unbiased estimator (Robert and Casella, 2013), in practice millions of runs are needed to drive down the variance of the estimator.

Multi-fidelity methods seek to drive down the sampling cost for reliability estimation through the introduction of thriftier models for data generation. A multi-level MC method (Cliffe et al., 2011; Giles, 2008) uses the variance reduction framework of control variates to combine the outputs of a hierarchy of coarse grid solutions. Litvinenko et al. (2013) used a residual-based indicator to decide when to evaluate a surrogate model versus the high-fidelity model for the purpose of estimating high-fidelity model statistics. However, these methods do not focus on the speedup of estimating failure probability. Other multi-fidelity approaches seek to adaptively trade-off between sampling the high-fidelity and surrogate model (Li and Xiu, 2010; Li et al., 2011). For a more comprehensive review of different methods see Peherstorfer et al. (2018b) and Fernández-Godino et al. (2016).

GPs are often applied as surrogates with the goal of estimating failure probabilities because of their common application in sequential learning described in Section 2.3.3. Bect et al. (2012) define the goal of minimizing Bayesian risk with their estimator $\hat{\alpha}$ through stepwise uncertainty reduction (SUR). In practice they estimate an upper bound on the

risk and use quadrature to estimate the integration. Many other methods seek to achieve a similar estimate through contour finding.

2.4.1 Contour Estimation

A large focus of estimating failure probability is using contour estimation to find the boundary surrounding an area of interest (such as a failure region). Other terms for contours include level and excursion sets. A failure region's contour can be expressed as the set

$$\mathcal{C} = \{\mathbf{x} \in \mathcal{X} : g(\mathcal{T}(\mathbf{x})) = 0\}. \tag{2.27}$$

In most problems related to contour estimation, the model $\mathcal{T}(\cdot)$ produces a one-dimensional output y. For simplicity, the contour can also be defined through a target value T,

$$\mathcal{C} = \{\mathbf{x} \in \mathcal{X} : \mathcal{T}(\mathbf{x}) = T\}. \tag{2.28}$$

GPs are a common modeling tool used for contour estimation due to their ability to facilitate sequential learning. Bryan and Schneider (2008) use sequential design based on balancing information gain near the contour with predictive variance. Their work is extended to use a composite function of multiple outputs to find a single contour. In Gotovos (2013), sequential selection seeks to reduce the ambiguity of the classification of points near the contour. Here they introduce modifications for a contour based on percentage of the maxima instead of a fixed value and batch selection. Ranjan et al. (2008) proposed a variation of Expected Improvement (Jones et al., 1998) that seeks to sample on the contour where the variance is largest. Another modification to EI calculates an expected feasibility function that is only integrated within a window of the contour threshold (Bichon et al., 2008).

The SUR approach is also adapted for contour finding. Chevalier et al. (2014) extend SUR to finding a level set with batch selection. They include a closed-form expression for one of the integrals in SUR, but still rely on estimation of another integral. Bayesian subset selection (Bect et al., 2017) ports the stepwise nature of SUR through estimation of a series of contours that decrease in volume. Work by Hu and Ludkovski (2017) use a multi-surface ranking approach with a Bayesian optimization flavor.

Picheny et al. (2010) introduced a weighted IMSE criterion for sequential design using a GP. They apply a Gaussian kernel weight on the distance of the proposed new point's predicted response to the target T,

$$W(\mathbf{x}) = \frac{1}{\sqrt{2\pi(\sigma_\epsilon^2 + \sigma_N^2(\mathbf{x}))}} \exp\left(-\frac{(\mu_N(\mathbf{x}) - T)^2}{2(\sigma_\epsilon^2 + \sigma_N^2(\mathbf{x}))}\right), \tag{2.29}$$

with σ_ϵ^2 being a hyperparameter suggested by the authors to be set at 5% of the output's range. Due to the burden of integration, they also developed the simpler criterion of weighted predictive variance (referred to as tMSE) with equation (2.29).

The strategy of using entropy as an acquisition function approaches contour finding as a classification problem. By defining discrete events w_i for $i = 1, \cdots, k$ based on the response's relation to the contour (i.e. above or below), the following definition of entropy can be used:

$$H(W) = -\sum_{i=1}^{k} \mathbb{P}(w_i) \log \mathbb{P}(w_i). \tag{2.30}$$

Oakley (2004) use equation (2.30) to select a entire LHS design at once by calculating the entropy for a total design. This method is not exactly aligned with adaptive design and was applied to quantiles near 95%. Contour Location Via Entropy Reduction (CLoVER; Marques et al., 2018) greedily maximizes the reduction in contour entropy (based on equation (2.30)) through a lookahead scheme. This acquisition function includes an integral that must

be estimated with a series of reference points or knots, similar to implementations of SUR and tIMSE.

Lyu et al. (2018) proposed several new acquisition functions inspired by earlier works. Maximum contour uncertainty (MCU) employs an upper confidence bound for Bayesian optimization Srinivas et al. (2012). MCU combines the goal of minimizing the predictive surface with maximizing posterior uncertainty. Another expression, maximal empirical error (MEE), measures the local probability of misclassification. MEE contains the same spirit of Bichon et al. (2008), Ranjan et al. (2008), Echard et al. (2010), and Bect et al. (2012), wanting to sample where the response is above a threshold with high uncertainty. Lyu et al. (2018) modify the SUR method (Bect et al., 2012) to contain the focus for contour finding with one step look-ahead. As an extension to cSUR, they also define integrated contour uncertainty (ICU), which targets global reduction in predictive uncertainty. The authors provide valuable comparisons with these methods for noisy data with standard GPs and student-t processes TPs), finding ICU performs the best in problems of high-dimension with misspeficied noise.

2.4.2 Importance Sampling

Another class of methods that aim to improve upon the number of model evaluations needed to estimate a statistic such as failure probability is that of *importance sampling* (Srinivasan, 2013). Techniques within this class seek to reduce an estimator's variance by proposing clever strategies to sample from distributions that cannot be sampled directly. Other applications of importance sampling are described in Robert and Casella (2013) and Srinivasan (2013).

The fundamental idea of importance sampling it to generate a bias distribution ($\mathbb{F}_\star$) that is easy to draw samples from that are then weighted to reflect the ratio of densities between

the bias distribution and distribution of interest ($\mathbb{F}$). By introducing a bias distribution $\mathbb{F}_\star$ with density $f_\star$, the failure probability can be expressed as

$$\begin{aligned}\alpha &= \mathbb{E}[\mathbb{1}_{g(\mathcal{T}(X))>0}] \\ &= \int_{\mathbf{x}\in\Omega} \mathbb{1}_{g(\mathcal{T}(\mathbf{x}))>0} \frac{f(\mathbf{x})}{f_*(\mathbf{x})} f_*(\mathbf{x}) dx \\ &= \int_{\mathbf{x}\in\Omega} \mathbb{1}_{g(\mathcal{T}(\mathbf{x}))>0} W(\mathbf{x}) d\mathbb{F}_* \\ &= \mathbb{E}_*[\mathbb{1}_{g(\mathcal{T}(X))>0} W(X)],\end{aligned} \tag{2.31}$$

where

$$W(\cdot) = \frac{f(\cdot)}{f_*(\cdot)} \tag{2.32}$$

is the weight of true input density to biased density. Equation (2.31) translates to the following Monte Carlo estimate based on samples from $\mathbb{F}_*$,

$$\hat{\alpha}_{IS} = \frac{1}{\acute{N}} \sum_{i=1}^{\acute{N}} \mathbb{1}_{g(\mathcal{T}(\mathbf{x}_i))>0} W(\mathbf{x}_i), \qquad \mathbf{x}_i \sim \mathbb{F}_*. \tag{2.33}$$

Biased densities can be generated by a direct transformation on the original random variables or through a family of densities. In the case of estimating failure probability, the desire is to define densities biased toward the failure region(s). With importance sampling techniques, MC samples can be generated that produce many more failure events, reducing the estimator's variance.

Different techniques exist to fit bias distributions. The cross-entropy method sequentially fits a distribution with respect to the Kullback-Leibler divergence to the inputs that produce failures in the system (De Boer et al., 2005; Rubinstein, 1997) . Another strategy, population Monte Carlo and its variations, seek to use an adaptive strategy to update weights on samples through a Markov transition kernel (Cappé et al., 2004; Douc et al., 2007a,b).

Cappé et al. (2008) developed a more flexible version of population Monte Carlo that allows for updating the kernels at each iteration.

Peherstorfer et al. (2016) introduced multifidelity importance sampling (MFIS) as a bridge between multifidelity methods with a biased surrogate and the asymptotically unbiased methodology of importance sampling. They rely upon a trained surrogate model to generate data and then classify series of inputs corresponding to the limit state function. The set of inputs corresponding to failures is used to train a Gaussian mixture model to serve as the biasing distribution $\mathbb{F}_\star$. Then, as is standard importance sampling procedure, a new set of inputs is generated by drawing from $\mathbb{F}_\star$. These inputs are used to evaluate the high-fidelity model and along with weighting and classification from the limit state function, and a failure probability estimate is produced. MFIS also benefits from using the asymptotic properties of importance sampling to provide error and cost analysis (Peherstorfer et al., 2016).

Chapter 3

Locally induced Gaussian processes

3.1 Introduction

Advancements and expansion of access to supercomputing, algorithms for finite element analysis, particle transport and agent-based modeling combine in modern times to produce simulation data of an unprecedented magnitude. Yet as modeling fidelity and configuration spaces continue to grow, coverage of representative cases is still sparse. GP regression is a common choice to fill in those gaps, emulating or serving as a surrogate for the data-generating mechanism. GP surrogates excel at downstream tasks from optimization to sensitivity analysis due to their out-of-sample predictive accuracy and uncertainty quantification (UQ) capability, and ability to interpolate the response when simulations are deterministic. For a review of computer experiments and surrogate modeling, see Santner et al. (2018) or (Gramacy, 2020).

However, GP inference and prediction calculations scale poorly for large data sets. GPs involve working with a multivariate normal (MVN) distribution whose dimension matches the training data $(\mathbf{X}_N, \mathbf{Y}_N)$ size, N. Matrix decomposition for covariance determinant and inverses is cubic in N. In practice, this means limiting N to the thousands – small by modern standards.

Work from across disciplines where GPs play a fundamental role (machine learning, geo-

statistics, computer experiments) targets remedies through various approximations. Some methods induce sparsity in the covariance (Aune et al., 2014; Gardner et al., 2018b; Pleiss et al., 2018; Solin and Särkkä, 2020; Titsias, 2009; Wilson and Nickisch, 2015) or precision matrix (Datta et al., 2016; Katzfuss and Guinness, 2021). Others propose divvying up the design space (Gramacy and Lee, 2008; Kim et al., 2005) and constructing multiple GPs by divide-and-conquer. Partitioning offers the potential for parallelized multicore computation, productively engaging untapped resources. It also induces statistical independence which can enhance flexibility when response surfaces have regime changes or exhibit other non-stationary behavior.

One framework, developed separately as *pseudo-inputs* in machine learning (e.g., Snelson and Ghahramani, 2006) and *predictive processes* in geostatistics (e.g., Banerjee et al., 2008), offers a low-rank approximation. Together, these two ideas are more recently referred to as *inducing point* methods. Rather than measuring covariances between all pairs of N training data points directly, a smaller reference set $\mathbf{\Psi}_M$ of $M \ll N$ inducing points or "knots" is used. Woodbury matrix identities make decompositions cubic in M, potentially dramatic savings. While space-filling works well, optimizing the multitude M and location of knots is fraught with challenges (e.g., Garton et al., 2020).

One thing that sets surrogate modeling of computer simulations apart from machine learning and geostats applications of GPs – beside time being of the essence – is an all-but-total emphasis on prediction and UQ above other inferential tasks. This opens up new opportunities for computational and statistical economies by taking a *transductive* approach to learning (Vapnik, 2013): let the testing data dictate how training is done. Accurate, approximate GP prediction at an input $\mathbf{x}'$ can be based on a subset of data nearby $\mathbf{x}'$, leading to the so-called local approximate GP (LAGP; Gramacy and Apley, 2015). Small data subsets $n \ll N$ mean faster matrix decomposition, and potential for embarrassingly parallel

implementation (Gramacy et al., 2014), through an infinite divide-and-conquer/partition scheme.

The best sub-designs for predicting at $\mathbf{x}'$ depend on the training data $\mathbf{X}_n(\mathbf{x}') \subset \mathbf{X}_N$ nearby $\mathbf{x}'$. Those which are the very closest – a nearest neighbor (NN) subset – may not be ideal for all predictive goals, such as minimizing mean-squared prediction error (MSE; Stein et al., 2004; Vecchia, 1988). Best results require sequentially optimizing a criterion for each $\mathbf{x}'$ to greedily build $\mathbf{X}_n(\mathbf{x}')$. Although speedy and vastly parallelizable, handling N in the millions in a matter of minutes, it can still represent a substantial computational effort, growing cubically with n and combinatorially in $\binom{N}{n}$ choices. Authors have long opined that novel searches for each $\mathbf{x}' \in \mathbf{X}$ could be short-cut by learning some kind of re-locatable *template* of local sub-design characteristics (Gramacy and Haaland, 2016; Sung et al., 2018). However, a truly thrifty scheme has so far remained elusive.

A potential answer may lie in hybridizing inducing point and local GP schemes – a variation on a recently popular theme of combining sparse GP methods with local models (Liu et al., 2019; Tan et al., 2016). The basic idea is as follows: search locally for m inducing points $\mathbf{\Psi}_m(\mathbf{x}')$ in order to predict nearby $\mathbf{x}'$, specifically on a NN set $\mathbf{X}_n(\mathbf{x}')$. Having $m \ll n \ll N$ leads to a manageable cascade of calculations. I show how greedy optimization of $\mathbf{\Psi}_m(\mathbf{x}')$, via a closed form weighted integrated MSE (wIMSE) criterion and gradients, avoids combinatorial sub-design search. Moreover, $\mathbf{\Psi}_m(\mathbf{x}')$ can be used as a template, relocated anywhere for any $\mathbf{x}'$ without re-optimization. In fact, I show that even locally space-filling schemes make for adequate templates in this setting. The result is a *locally induced GP (LIGP)* approximation which is nearly as accurate as LAGP, sometimes even more accurate, and is faster. Whereas LAGP was limited by small-n neighborhoods regardless of what the data prefer, I show that LIGP is not. I explore neighborhoods more than double the size of LAGP and demonstrate accuracy improvements for commensurate computational

effort. This allows the user, for the first time, to fully explore the statistical–computational efficiency Pareto frontier in the context of local GP approximation.

The remainder of the chapter is organized as follows. Section 3.2 provides an overview of GP regression and various scalable models, including local and inducing points methods by way of motivating my hybrid approach. Section 3.3 describes the joining of local and inducing points methods comprising LIGP. I detail some refinements to LIGP, including local inducing point templates, in Section 3.4. Illustrative examples are provided throughout, however Section 3.5 offers a systematic comparison of LIGP and LAGP variations to using both synthetic and real benchmark examples. Section 3.6 concludes with a discussion.

3.2 Foundations in Gaussian process approximation

Here, I highlight relevant surrogate modeling and scalable GP methods and provide motivation for a new criterion for placement of global and local inducing points.

3.2.1 Gaussian process regression

Consider an unknown function $z : \mathbf{X}_N \subset \mathbb{R}^d \to \mathbb{R}$ for a set of d-dimensional design locations $\mathbf{X}_N = (\mathbf{x}_1, ..., \mathbf{x}_N)^\top$ and corresponding observations $\mathbf{Y}_N = (y_1, ..., y_N)^\top$. GPs are common surrogates for such data (Sacks et al., 1989b), especially as arising from deterministic computer simulations $z(\cdot)$, and boil down to placing an MVN prior on the observations $\mathbf{Y}_N$. Gaussians are uniquely defined by a mean vector, which I take as zero for simplicity, and an $N \times N$ covariance matrix $\mathbf{K}_N$. The joint model for all responses is $\mathbf{Y}_N \sim \mathcal{N}_N(\mathbf{0}, \tau^2(\mathbf{K}_N + \epsilon_K \mathbb{I}_N))$ where τ^2 is a scale hyperparameter and $\mathbf{K}_N$ is comprised of entries based on a kernel $k_\theta(\mathbf{x}_i, \mathbf{x}_j)$. The jitter parameter ϵ_K is set as small as possible (for

interpolating deterministic simulations) while maintaining well-conditioned positive-definite covariances (Neal, 1998), and $\mathbb{I}_N$ denotes an $N \times N$ identity matrix. This presentation is agnostic to the choice of $k_\theta(\cdot, \cdot)$ except that it be based on inverse distances in the input space. My empirical work favors a squared exponential kernel with lengthscale hyperparameter θ, initially defined in equation (2.5). The entry in the ith row and j^{th} column of $\mathbf{K}_N$ is defined as

$$\mathbf{K}_N^{ij} = k_\theta(\mathbf{x}_i, \mathbf{x}_j) = \exp\left\{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\theta}\right\}. \tag{3.1}$$

Other common kernels include the Matérn family (Gramacy, 2020; Stein, 2012, Section 5.3.3).

Inference for unknown hyperparameters (θ, τ^2) can proceed by maximum likelihood estimation through the log MVN pdf and its closed-form derivatives. Some hyperparameters, like $\hat{\tau^2} = N^{-1}\mathbf{Y}_N^\top \mathbf{K}_N^{-1}\mathbf{Y}_N$, have tidy expressions conditional on others, like θ, which must be optimized numerically. Since MVN pdfs involve $|\mathbf{K}_N|$ and $\mathbf{K}_N^{-1}$, computation is on the order of $\mathcal{O}(N^3)$, limiting training data sizes N to the small thousands on most desktop machines. In custom setups with highly distributed architectures, stochastic approximations based on linear conjugate gradients and Lanczos quadrature can push those boundaries (Gardner et al., 2018a; Ubaru et al., 2017; Wang et al., 2019).

For fixed hyperparameters $(\hat{\tau^2}, \hat{\theta})$, a predictive distribution for $Y(\mathbf{x}')$ arises as standard MVN conditioning via an $(N+1)$-dimensional MVN for $(Y(\mathbf{x}'), \mathbf{Y}_N)$. By factoring out τ^2 from the equations in (2.2), the moments of that Gaussian distribution are:

$$\begin{aligned} \mu_N(\mathbf{x}') &= \mathbb{E}(Y(\mathbf{x}^*) \mid \mathbf{Y}_N) = \mathbf{k}_N^\top(\mathbf{x}')\mathbf{K}_N^{-1}\mathbf{Y}_N \\ \sigma_N^2(\mathbf{x}') &= \mathbb{V}\text{ar}(Y(\mathbf{x}') \mid \mathbf{Y}_N) = \hat{\tau^2}\left(k_\theta(\mathbf{x}', \mathbf{x}') - \mathbf{k}_N^\top(\mathbf{x}')\mathbf{K}_N^{-1}\mathbf{k}_N(\mathbf{x}')\right), \end{aligned} \tag{3.2}$$

where $\mathbf{k}_N(\mathbf{x}') = (k_\theta(\mathbf{x}', \mathbf{x}_1), ..., k_\theta(\mathbf{x}', \mathbf{x}_N))^\top$. These calculations are also in $\mathcal{O}(N^3)$, although again linear algebra tricks can mitigate that to an extent.

3.2.2 Inducing points

A more direct approach to speedy GP approximation in the face of big N is to impose a low-rank structure on covariance. The idea originated with local data subsets for splines (Poggio and Girosi, 1990; Wahba, 1990), and later was applied to GPs (Csató and Opper, 2002; Seeger et al., 2003; Smola and Bartlett, 2001). Snelson and Ghahramani (2006) proposed that these reference locations not be restricted to a subset of the data. First attempts at a unifying perspective for sparse approximate GPs were made by Quiñonero and Rasmussen (2005) and Rasmussen and Williams (2006, Chapter 8), with the former referring to these latent reference variables as *inducing inputs*. Outside of the machine learning community, Banerjee et al. (2008) applied similar techniques to develop *predictive processes*. Here, I adopt a big-tent inducing points nomenclature.

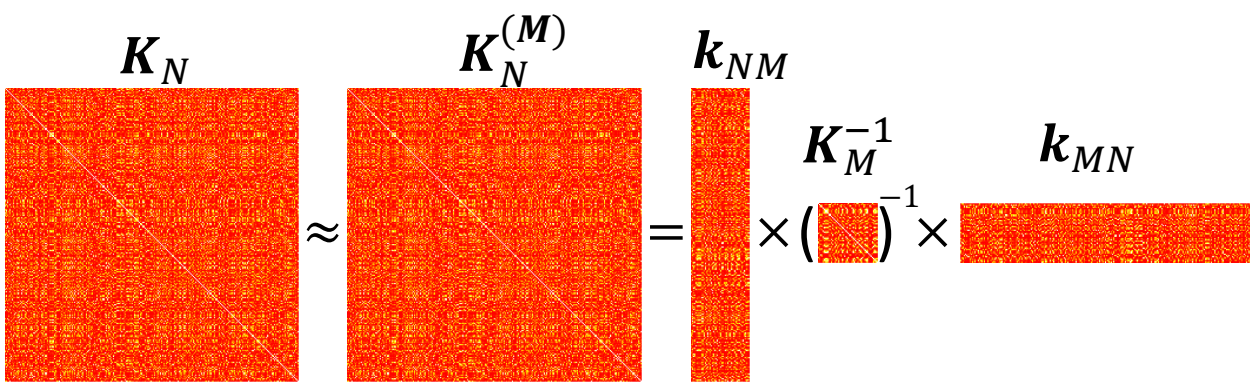


Figure 3.1: Heat map diagram of the Nyström approximation of $\mathbf{K}_N$ through M inducing points. Rank-M approximation $\mathbf{K}_N^{(M)}$ is strikingly similar to full rank $\mathbf{K}_N$.

Let $\mathbf{\Psi}_M = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_M)^\top$ be M inducing points in the same space as $\mathbf{X}_N$, but they need not coincide with any elements of $\mathbf{X}_N$. Notate $\mathbf{K}_M$ as a kernel matrix built from $\mathbf{\Psi}_M$ and $k_\theta(\cdot, \cdot)$, e.g., in equation (3.1); similarly, write $\mathbf{k}_{NM}$ as cross evaluations of the kernel between $\mathbf{X}_N$ and $\mathbf{\Psi}_M$. Most variations on inducing point methods base GP approximations on the so-called Nyström approximation (Williams and Seeger, 2001): $\mathbf{K}_N \approx \mathbf{K}_N^{(M)} = \mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top$.

See Figure 3.1. Rather than calculate covariance between all pairs in $\mathbf{X}_N$, instead use $M \ll N$ references $\mathbf{\Psi}_M$ to induce a similar structure $\mathbf{K}_N^{(M)}$.

Snelson and Ghahramani (2006) introduced a diagonal correction on the Nyström approximation

$$\begin{aligned}\mathbf{\Sigma}_N^{(M)} &= \tau^2(\mathbf{K}_N^{(M)} + \epsilon_K \mathbb{I}_N) \\ &= \tau^2\Big(\mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top + \Delta_N^{(M)} + \epsilon_K \mathbb{I}_N\Big),\end{aligned} \tag{3.3}$$

where $\Delta_N^{(M)} = \text{Diag}\{\mathbf{K}_N - \mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top\}$. This ensures that $\mathbf{K}_N^{(M)}$ and $\mathbf{K}_N$ contain the same diagonal elements so that when $\mathbf{\Psi}_M \equiv \mathbf{X}_N$, $\mathbf{\Sigma}_N^{(M)}$ in equation (3.3) reduces to the standard GP covariance $\mathbf{\Sigma}_N = \tau^2(\mathbf{K}_N + \epsilon_K \mathbb{I}_N)$. Both approximations allow for decomposition of $\mathbf{\Sigma}_N^{(M)}$ through Woodbury matrix identities (Harville, 1998) as in equation (2.10). Assuming a deterministic model, define $\mathbf{Q}_M^{(N)} = \mathbf{K}_M + \mathbf{k}_{NM}^\top \Omega_N^{-1(M)}\mathbf{k}_{NM} + \epsilon_Q \mathbb{I}_M$ and $\Omega_N^{(M)} = \Delta_N^{(M)} + \epsilon_K \mathbb{I}_N$ to include jitter terms for matrix inversion. Since $\Omega_N^{(M)}$ is an $N \times N$ diagonal matrix and can be stored and manipulated as a vector, I elect to not embolden its notation like that of other matrices. Hyperparameter inference is achieved by maximizing the logarithm of the MVN likelihood $\mathbf{Y}_N \sim \mathcal{N}\left(\mathbf{0}, \tau^2(\mathbf{k}_{NM}\mathbf{K}_M^{-1}\mathbf{k}_{NM}^\top + \Omega_N^{(M)})\right)$ in equation (2.11). Differentiating equation (2.11) with respect to τ^2 and solving yields the closed-form estimate

$$\hat{\tau^2}^{(N,M)} = N^{-1}\mathbf{Y}_N^\top\left(\Omega_N^{-1(M)} - \Omega_N^{-1(M)}\mathbf{k}_{NM}\mathbf{Q}_M^{-1(N)}\mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\right)\mathbf{Y}_N. \tag{3.4}$$

There is not a similar closed-form solution for the lengthscale. Numerical solvers like `optim` in R can work with negative concentrated log-likelihood

$$\begin{aligned}-\ell\left(\mathbf{X}, \mathbf{Y}, \mathbf{\Psi}_M, \theta\right) \propto N \log\Big(\mathbf{Y}_N^\top\left(\Omega_N^{-1(M)} - \Omega_N^{-1(M)}\mathbf{k}_{NM}\mathbf{Q}_M^{-1(N)}\mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\right)\mathbf{Y}_N\Big) \\ + \log|\mathbf{Q}_M^{(N)}| - \log|\mathbf{K}_M| + \mathbf{1}_N^\top \log(\Omega_N^{(M)})\mathbf{1}_N,\end{aligned} \tag{3.5}$$

and closed form derivatives (not shown) to obtain $\hat{\theta}^{(N,M)}$. In practice this works well because the surfaces are either convex in hyperparameters, or are nearly so.

Analogues to equations (3.4–3.5) reduce full rank prediction from $\mathcal{O}(N^3)$ down to $\mathcal{O}(NM^2)$ flops. Following equation (3.3), predictive equations are Gaussian with

$$\begin{aligned} \mu_{M,N}(\mathbf{x}') &= \mathbf{k}_M^\top(\mathbf{x}')\mathbf{Q}_M^{-1(N)}\mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\mathbf{Y}_N \\ \sigma^2_{M,N}(\mathbf{x}') &= \tau^2\Big(k_\theta(\mathbf{x}',\mathbf{x}') - \mathbf{k}_M^\top(\mathbf{x}')\left(\mathbf{K}_M^{-1} - \mathbf{Q}_M^{-1(N)}\right)\mathbf{k}_M(\mathbf{x}')\Big)\,, \end{aligned} \tag{3.6}$$

where $\mathbf{k}_M(\mathbf{x}') = k_\theta(\mathbf{\Psi}_M, \mathbf{x}')$. When optimizing $\mathbf{\Psi}_M$ via log likelihood, the value $\mathbf{Q}_M^{-1(N)}\mathbf{k}_{NM}^\top\Omega_N^{-1(M)}\mathbf{Y}_N$ can be re-used from $\mathbf{\Sigma}_N^{-1(M)}\mathbf{Y}_N$. Thus, prediction requires only $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ additional flops compared to $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ for a full GP model.

3.2.3 Optimal induction

Suppose, for now, that the number of inducing points M is fixed by computational limitations. Snelson and Ghahramani (2006) suggested selecting locations $\mathbf{\Psi}_M$ through the marginal log-likelihood. Such a strategy is prone to overfitting (Bauer et al., 2016), while the Variational Free Energy (VFE) approximation—a lower bound on the marginal likelihood—is not (Hoffman et al., 2013; Titsias, 2009). Yet even with VFE's variational construction of the likelihood, its optimization still requires a cubic cost on a highly-multimodal surface (Bauer et al., 2016), which I explore momentarily (see Figure 3.2). This begs the question if likelihood optimization is worth it over relatively convenient space-filling options.

Methods for "choosing inputs," known more widely as statistical design or *active learning*, have potential to reduce the cost of selecting inducing points. A slew of acquisition functions for greedy design point selection is available for such diverse goals as integral es-

timation (Fernández et al., 2020; Kanagawa and Hennig, 2019), space-fillingness (Busby, 2009; Svendsen et al., 2020), and posterior density approximation (Wang and Li, 2018). I propose that other variance-based (3.2) criteria, whose lesser (quadratic) computational cost more squarely targets predictive goals in surrogate modeling, may be appropriated for the selection of inducing points $\mathbf{\Psi}_M$ by routing through equation (3.6) instead.

In particular, I consider variations on integrated mean-squared error (IMSE) over a domain $\mathcal{X}$, with smaller being better:

$$\text{(IMSE)} \qquad I = \int_{\tilde{\mathbf{x}} \in \mathcal{X}} \sigma^2(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}}.$$

Choose $\sigma^2(\cdot) \equiv \sigma_N^2(\cdot)/\tau^2$ from equation (3.2), and I may be used to optimize the N coordinates of $\mathbf{X}_N$, or to choose the next ($N+1^{\text{st}}$) one ($\tilde{\mathbf{x}}_{N+1}$) in a sequential setting.[1] Closed form expressions are available for rectangular $\mathcal{X}$ and common kernels (e.g., Anagnostopoulos and Gramacy, 2013; Ankenman et al., 2010; Burnaev and Panov, 2015; Leatherman et al., 2018). Analytic derivatives $\frac{\partial I}{\partial \tilde{\mathbf{x}}_{N+1}}$ facilitate numerical optimization (Binois et al., 2018b; Gramacy, 2020, Chapters 4 & 10). Approximations are common otherwise (Gauthier and Pronzato, 2014; Gorodetsky and Marzouk, 2016; Gramacy and Lee, 2009; Pratola et al., 2017).

An analogue active learning heuristic from (Cohn, 1994), dubbed ALC, instead targets variance aggregated over a discrete reference set $\mathcal{X}$, originally for neural network surrogates:

$$\text{(ALC)} \qquad \Delta\sigma^2 = \sum_{\tilde{\mathbf{x}} \in \mathcal{X}} \sigma^2(\tilde{\mathbf{x}}) - \sigma^2_{\text{new}}(\tilde{\mathbf{x}}).$$

Seo et al. (2000) ported ALC to GPs taking $\sigma^2(\cdot) = \sigma_N^2(\cdot)$ and $\sigma^2_{\text{new}}(\cdot) \equiv \sigma^2_{N+1}(\cdot)$. If discrete and volume-based $\mathcal{X}$ are similar, then $\Delta\sigma^2 \approx c - I$, where c is constant on $\mathbf{x}_{N+1}$. Discrete

[1] Dividing out τ^2 removes dependence on $\mathbf{Y}$-values through $\hat{\tau^2}$. Greedy build-up of $\mathbf{x}_{n+1}$ over $n = N_0, \ldots, N-1$ is near optimal due to a supermartingale property (Bect et al., 2016).

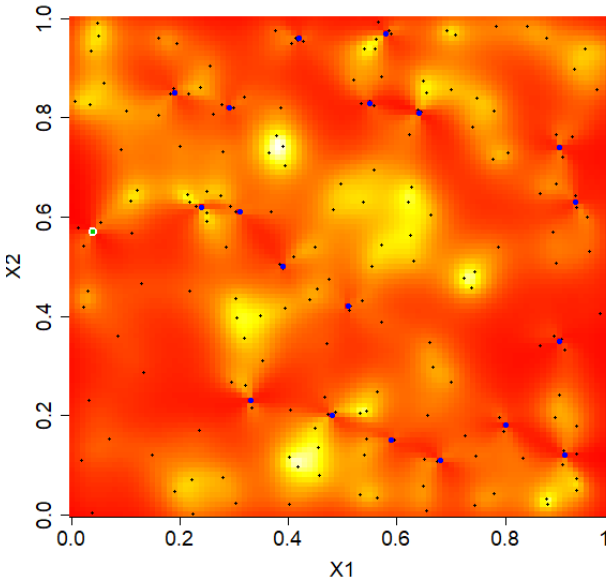
$\Delta\sigma^2$ via ALC is advantageous in transductive learning settings (Vapnik, 2013), where $\mathcal{X}$ can be matched with a testing set. Otherwise, analytic I via IMSE may be preferred.

Against that backdrop, I propose employing ALC and IMSE to select inducing points $\boldsymbol{\Psi}_M$. To my knowledge, using such variance-based criteria is novel in the literature on the selection of inducing points. The criteria below are framed sequentially, for an $M+1^{\text{st}}$ point given M collected already. Although I prefer this greedy approach – optimizing d coordinates one-at-a-time rather than Md all at once in a surface with many equivalent locally optimal configurations due to label-switching – either criteria is easily re-purposed for an all-at-once application. Under the diagonal-corrected Nyström approximation (3.3) and assuming coded $\mathcal{X} = [0,1]^d$,

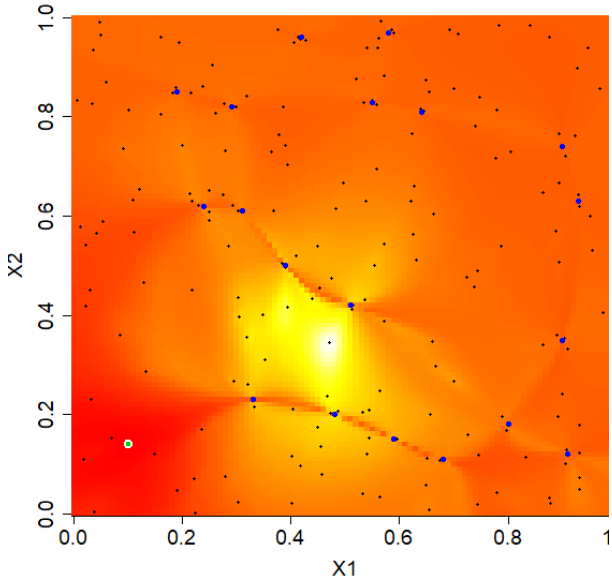
$$\begin{aligned} \text{ALC}_N^{(M+1)} &= \text{ALC}(\boldsymbol{\psi}_{M+1}; \mathbf{X}_N, \mathbf{Y}_N, \mathcal{X}, \boldsymbol{\Psi}_M) = c - \sum_{\tilde{\mathbf{x}} \in \mathcal{X}} \sigma^2_{M+1,N}(\tilde{\mathbf{x}}), \quad \text{and} \\ \text{IMSE}_N^{(M+1)} &= \text{IMSE}(\boldsymbol{\psi}_{M+1}, \mathbf{X}_N, \mathbf{Y}_N, \mathcal{X}, \boldsymbol{\Psi}_M) = E - \text{tr}\Big\{\Big(\mathbf{K}_{M+1}^{-1} - \mathbf{Q}_{M+1}^{-1(N)}\Big)\mathbf{W}_{M+1}\Big\}, \end{aligned} \tag{3.7}$$

where $E = \int_{\tilde{\mathbf{x}} \in \mathcal{X}} k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) d\tilde{\mathbf{x}}$ and $\mathbf{W}_{M+1}$ is $(M+1) \times (M+1)$ via $w(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j) = \int_{\tilde{\mathbf{x}} \in \mathcal{X}} k(\boldsymbol{\psi}_i, \tilde{\mathbf{x}}) k(\boldsymbol{\psi}_j, \tilde{\mathbf{x}}) d\tilde{\mathbf{x}}$ for $i, j \in \{1, ..., M+1\}$.

To explore inducing point optimization, I use a toy 2d problem with the equation $z(x_1, x_2) = x_1 \exp\{-x_1^2 - x_2^2\}$ for $x_1, x_2 \in [-2, 4]$. Figure 3.2 shows variational lower-bound of the log-likelihood (left) and ALC/IMSE surfaces (right) for $\boldsymbol{\psi}_{20}$ given a modestly sized training data set $(\mathbf{X}_N, \mathbf{Y}_N)$ of size $N = 200$. Similarities in the two surfaces are apparent. Many low/red areas coincide, but the optimizing locations (green dots), found via multi-start local optimization with identically fixed kernel hyperparameters, do not. Even after taking great care to humbly restrict searchers, e.g., from crossing $\boldsymbol{\Psi}_M$ locations, sometimes upward of 1000 evaluations were required to achieve convergence. Consequently, quadratic ALC/IMSE is faster.



(a) Variational lower bound of log-likelihood surface



(b) ALC/IMSE surface

Figure 3.2: In both panels: $N = 200$ training data points (black dots) and $M = 19$ inducing points (blue dots), selecting the twentieth one (green) by two criteria: (a) variational lower bound of the log-likelihood; (b) ALC/IMSE. Yellow is higher/red lower.

In a similar experiment, I sought to compare the predictive accuracy of sparse GP models with inducing points selected sequentially with VFE, IMSE, and ALC to a full GP. Figure 3.3 compares the three methods to themselves and to a full GP over $M = 1, \ldots, 100$ tracking root MSE (RMSE) via Monte Carlo (MC) averaging over training $\mathbf{X}_N$ and testing $\mathcal{X}$ locations. To manage the computational cost of evaluating criteria on a dense grid, training data sizes were limited to $N = 100$.[2] Observe that all three methods offer a decent approximation to the full GP with close to 85 inducing points. Zoomed boxplots (upper-right panel) show that ALC is consistently best. If you know where you are going to be tested, you should "design" your $\mathbf{\Psi}_M$ to focus there. If you do not, then you are (eventually) next-best by integrating over the input domain with IMSE. VFE performs worst because likelihood is

[2]Ordinary IMSE was used, substituting inducing points in for design points, as described in (Binois et al., 2018b). Progress is blocky because individual inducing point additions do not substantial alter space-filling properties until most of a "new row" of sites are added in this 2d example.

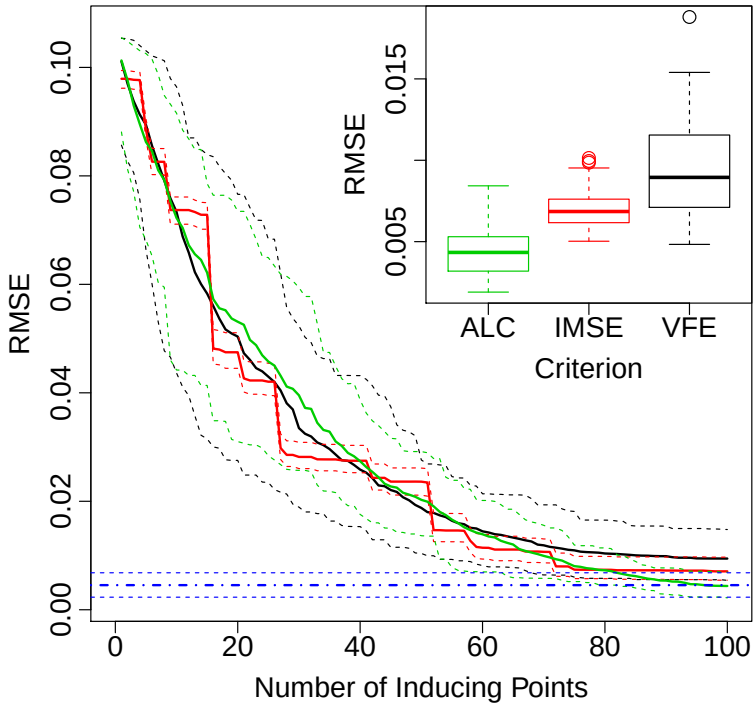


Figure 3.3: Approximate GP performance via RMSE and number of inducing points, M, compared to a full GP (blue). Means (solid) and central 90% intervals (dashed) arise from 30 replicates. Boxplots in the top right zoom in at $M = 100$.

imperfectly aligned to the RMSE criteria.

3.2.4 Local approximate Gaussian processes

Rather than massage the GP framework to cope with the entire data set at once, e.g., by working with a single global data subset, a *local approximate GP* (LAGP; Gramacy and Apley, 2015) considers disparate local data subsets depending on each of the predictive location(s) $\mathbf{x}'$ of interest. Such subsets can be much smaller because, under typical inverse-distance based correlation (3.1), training data inputs $\mathbf{X}_N$ far from each $\mathbf{x}'$ provide little added value to the underlying predictor. Specifically, suppose that $(\mathbf{X}_n(\mathbf{x}'), \mathbf{Y}_n(\mathbf{x}'))$ represents an n-sized subset, or *neighborhood* of the training data nearby $\mathbf{x}'$, e.g., comprised of nearest neighbors (NNs). Then, given a suitable hyperparameterization, prediction could follow

equation (3.2) using $(\mathbf{X}_n(\mathbf{x}'), \mathbf{Y}_n(\mathbf{x}'))$ rather than the full $(\mathbf{X}_N, \mathbf{Y}_N)$. This can potentially provide drastic computational savings when $n \ll N$, even though the calculations would still be cubic in n.

In this framework, the subset size n and neighborhood $\mathbf{X}_n(\mathbf{x}')$ must be determined. Because flops grow quickly with n, this value is usually fixed by computational limitations, just like the number of inducing points, M. A default in the `laGP` software (Gramacy, 2016) is $n = 50$, see Section 3.4.3 for further discussion. Fixing n, it turns out that NN subdesign, originally suggested by Emery (2009) in a 2d geostatistics setting, is sub-optimal by several criteria (Stein, 2012; Vecchia, 1988). However, exhaustively searching among all $\binom{N}{n}$ alternatives for each $\mathbf{x}'$ is combinatorially infeasible. Gramacy and Apley (2015) showed that greedy neighborhood selection via ALC approximately minimizes a MSE criteria common in surrogate modeling settings. Specifically, choose a singleton reference set $\mathcal{X} = \{\mathbf{x}'\}$, with $\sigma^2_{\text{new}}(\cdot) = \sigma^2_{n+1}(x)$ derived from $(\mathbf{X}_n(\mathbf{x}'), \mathbf{Y}_n(\mathbf{x}'))$ and select among $\mathbf{x}_{n+1} \in \mathbf{X}_N \setminus \mathbf{X}_n(\mathbf{x}')$ candidates.[3]

Care is taken to ensure computational demands in each update and ALC optimization do not exceed $\mathcal{O}(n^2)$ so that the entire scheme's flops are not worse in order than using NNs (i.e., cubic in n). For example, if $v_n(\mathbf{x}_{n+1}) = \mathbf{k}_{n+1}(\mathbf{x}_{n+1}, \mathbf{x}_{n+1}) - \mathbf{k}^\top_{n+1}(\mathbf{x}_{n+1})\mathbf{K}^{-1}_{n+1}\mathbf{k}_{n+1}(\mathbf{x}_{n+1})$ represents the kernel portion of $\sigma^2_n(\mathbf{x}_{n+1})$, then the change

$$\begin{aligned}\Delta v_n(\mathbf{x}') &= v_n(\mathbf{x}') - v_{n+1}(\mathbf{x}') \\ &= \mathbf{k}^\top_n(\mathbf{x}')\mathbf{h}_n(\mathbf{x}_{n+1})v_n(\mathbf{x}_{n+1})\mathbf{k}_n(\mathbf{x}') + 2\mathbf{k}^\top_n(\mathbf{x}')\mathbf{h}_n(\mathbf{x}_{n+1})k_\theta(\mathbf{x}_{n+1}, \mathbf{x}') \\ &\quad + k_\theta(\mathbf{x}_{n+1}, \mathbf{x}')^2/v_n(\mathbf{x}_{n+1}),\end{aligned} \tag{3.8}$$

can be updated in $\mathcal{O}(n^2)$ via partition inverse equations (Barnett, 1979) using $\mathbf{h}_j(\mathbf{x}_{n+1}) =$

[3]Here I am abusing notation a little to describe an inductive process $n \to n+1$ and referring to n as the final local design size as opposed to introducing a new iterator.

$\mathbf{h}_n(\mathbf{x}_{n+1})\mathbf{h}_n^\top(\mathbf{x}_{n+1})$, $\mathbf{h}_n(\mathbf{x}_{n+1}) = -\mathbf{K}_n^{-1}\mathbf{k}_n(\mathbf{x}_{n+1})/v_n(\mathbf{x}_{n+1})$.

Despite being massively parallelizable (Gramacy et al., 2014) for many $\mathbf{x}'$ and over candidates $\mathbf{x}_{n+1} \in \mathbf{X}_N \setminus \mathbf{X}_n(\mathbf{x}')$, further approximations are made in order to shortcut $\mathcal{O}(N)$ subroutines in an $\mathcal{O}(n^2)$ scanning over that set (Gramacy and Haaland, 2016; Sun et al., 2019a; Sung et al., 2018). Several groups of authors have suggested that it might be possible to design a "template" sub-design that could be applied automatically, after simple shifting/scaling for each $\mathbf{x}'$, without exhaustive search of $\mathbf{X}_N \setminus \mathbf{X}_n(\mathbf{x}')$. Non-uniform global designs $\mathbf{X}_N$ render this a non-starter. Sparse design coverage in some regions, and dense in others, demands bespoke calculation in each $\mathbf{x}'$ instance. Even with highly regular (e.g., gridded) global designs $\mathbf{X}_N$, local coverage can be irregular at the boundaries.

Local design topology is twinned with subset size, n. Accommodating wiggly test problems benefit with reactive dynamics offered by smaller n is easy, because that means faster execution. But n much larger than the default of $n = 50$ can be a deal-breaker on speed grounds regardless of accuracy boosts in less wiggly settings.

3.3 Inducing point neighborhoods

Inducing points offer computational savings, but several drawbacks remain. Predictive accuracy suffers when they are placed far from testing locations. Optimization by likelihood can perform worse than simple space-filling (Section 3.2.3). Computational costs are still cubic in a big number, despite $M \ll N$ because you need enough M to fill the input volume. Multi-processing parallel schemes via likelihood (Chen et al., 2013) and stochastic variational inference (Hensman et al., 2013; Hoang et al., 2015; Schürch et al., 2020) offer limited respite because they operate on the full data.

I thus propose a *locally induced GP (LIGP)* by hybridizing ordinary, "global" inducing point schemes with LAGP. This brings knock-on benefits to the local data-subsetting world: speed-ups, selection of neighborhood size (larger for smoother processes), long-elusive template schemes (Section 3.4). LIGP operates similarly to LAGP via neighborhoods $\mathbf{X}_n(\mathbf{x}') \subset \mathbf{X}_N$. If a greedy scheme like ALC is used to fill $\mathbf{X}_n(\mathbf{x}')$, it would include an exhaustive search on the order of $\mathcal{O}(Nn^3)$. Instead I choose simple NN approach, incurring an amortized one-off $\mathcal{O}(N \log N)$ cost. Effort is reallocated into choosing local inducing points $\mathbf{\Psi}_m(\mathbf{x}')$ for $\mathbf{X}_n(\mathbf{x}')$, which are free to take on any values, at cubic in m cost. My multiplicity notation is intended to convey $m \ll n \ll M \ll N$, although that hierarchy need not be strict. Small m allows wider local scope with bigger n without a substantial computational hit.

Algorithm 1 outlines the LIGP prediction algorithm, which can be run independently for each $\mathbf{x}' \in \mathbf{X}'$. For each $\mathbf{x}'$, a local neighborhood $\mathbf{X}_n(\mathbf{x}')$ is built from a NN subset of $\mathbf{X}_N$ followed by a set of inducing points $\mathbf{\Psi}_m(\mathbf{x}')$. Various methods to select $\mathbf{\Psi}_m(\mathbf{x}')$ are explored in the following sections.

Algorithm 1 LIGP Prediction

procedure LIGP.PRED(m, n, $\mathbf{X}'$,$\mathbf{X}_N$,$\mathbf{Y}_N$, $\mathcal{X}$)
  for $i = 1, \ldots, N' = |\mathbf{X}'|$ **do** ▷ Each $\mathbf{x}'_l \in \mathbf{X}'$, potentially in parallel
    $\{\mathbf{\Psi}_m, \mathbf{X}_n\} \leftarrow \mathrm{IP}(\ldots)$ ▷ Any of Algorithms 2–4
    $\mathbf{Y}_n \leftarrow Y(\mathbf{X}_n)$ ▷ Extract from $\mathbf{Y}_N$ at neighborhood
    $\hat{\tau^2}, \hat{\theta} \leftarrow \mathrm{argmax}_{\tau^2,\theta}\, \mathrm{LLik}(\tau^2, \theta, \mathbf{X}_n, \mathbf{Y}_n, \mathbf{\Psi}_m)$ ▷ Local MLE, Eqs. (3.4–3.5)
    $\{\hat{\mu}^{(i)}, \hat{\sigma}^{2(i)}\} \leftarrow$ GP.PRED($\mathbf{x}'_i \mid \mathbf{X}_n, \mathbf{Y}_n, \mathbf{\Psi}_m, \hat{\tau^2}, \hat{\theta}$) ▷ equation (3.6)
  end for
  return $\{\hat{\mu}^{(i)}, \hat{\sigma}^{2(i)}\}_{i=1}^{N'}$
end procedure

3.3.1 Sequential selection of local inducing points

Changing focus to local neighborhoods $\mathbf{X}_n(\mathbf{x}')$ warrants a second look at selection criteria for inducing points $\mathbf{\Psi}_m(\mathbf{x}')$. Likelihoods here are a mismatch to surrogate modeling and machine learning predictive goals. Instead, I follow the LAGP format of greedy optimization via MSE. Given the connection between inducing $\mathbf{\Psi}_m(\mathbf{x}')$ and actual training locations $\mathbf{X}_n(\mathbf{x}')$, emphasis on prediction at singleton $\mathbf{x}'$ has deleterious effects. I tried this: $\mathbf{\Psi}_m(\mathbf{x}')$ "pile up" around $\mathbf{x}'$ leading to poor estimates of local lengthscale and curvature. Instead, I suggest a locally weighted IMSE criterion.

Suppose I have $\mathbf{\Psi}_m(\mathbf{x}')$ already and wish to choose the next inducing point $\boldsymbol{\psi}_{m+1}(\mathbf{x}')$. Dependence on $\mathbf{x}'$ is implicit below, although I shall drop it from the expressions and simply write $\mathbf{X}_n$, $\mathbf{\Psi}_m$, $\boldsymbol{\psi}_{m+1}$, etc., in order to streamline the notation. I presume that the study region is a hyperrectangle $\mathcal{X} = [a_k, b_k]_{k=1}^d$. Rather than integrate uniformly over that domain, reproducing an ordinary global IMSE whose closed form slightly generalizes (Binois et al., 2018b), I weight the calculation by proximity to the predictive location $\mathbf{x}'$. Although this weighting scheme could be treated as a tuning parameter, I choose a Gaussian measure proportional to the Gaussian kernel $k_\theta(\cdot, \mathbf{x}')$ to facilitate a similar closed-form solution:

$$
\begin{aligned}
\mathrm{wIMSE}_n^{(m+1)}(\boldsymbol{\psi}_{m+1}, \mathbf{x}') \quad &\equiv \mathrm{wIMSE}(\boldsymbol{\psi}_{m+1}, \mathbf{X}_n, \mathbf{Y}_n, \mathcal{X}, \mathbf{\Psi}_m, \mathbf{x}') \qquad (3.9)\\
&= \int_{\tilde{\mathbf{x}} \in \mathcal{X}} k_\theta(\tilde{\mathbf{x}}, \mathbf{x}') \frac{\sigma^2_{m+1,n}(\tilde{\mathbf{x}})}{\tau^2} \, d\tilde{\mathbf{x}} \\
&= \prod_{k=1}^{D} \Bigg((1+\epsilon_K) \int_{a_k}^{b_k} k_\theta(\tilde{\mathbf{x}}_k, \mathbf{x}'_k) d\tilde{\mathbf{x}}_k - \int_{a_k}^{b_k} k_\theta(\tilde{\mathbf{x}}_k, \mathbf{x}'_k)^{1/2} k_\theta(\tilde{\mathbf{x}}_k, \mathbf{\Psi}_{m+1,k}) \\
&\qquad \times \left[\mathbf{K}_{m+1}^{-1} - \mathbf{Q}_{m+1}^{-1(n)} \right] k_\theta(\tilde{\mathbf{x}}_k, \mathbf{\Psi}_{m+1,k})^\top k_\theta(\tilde{\mathbf{x}}_k, \mathbf{x}'_k)^{1/2} \; d\tilde{\mathbf{x}}_k \Bigg) \\
&= \frac{\sqrt{\theta\pi}}{2} \prod_{k=1}^{d} \left(\mathrm{erf}\left\{ \frac{\mathbf{x}' - a_k}{\sqrt{\theta}} \right\} - \mathrm{erf}\left\{ \frac{\mathbf{x}' - b_k}{\sqrt{\theta}} \right\} \right) \\
&\qquad - \mathrm{tr}\left\{ \left(\mathbf{K}_{m+1}^{-1} - \mathbf{Q}_{m+1}^{-1(n)} \right) \mathbf{W}'_{m+1} \right\},
\end{aligned}
$$

where erf is the error Gaussian function and $\mathbf{W}'_{m+1} = \prod_{k=1}^{d} \mathbf{W}'_{m+1,k}$. The $(i,j)^{\text{th}}$ entry of $\mathbf{W}'_{m+1,k}$ is

$$
\begin{aligned}
w'^{(i,j)}_{m+1,k} &\equiv w_{m+1,k}(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j) \\
&= \int_{a_k}^{b_k} k_\theta(\tilde{\mathbf{x}}_k, \mathbf{x}'_k) k_\theta(\tilde{\mathbf{x}}_k, \boldsymbol{\psi}_{i,k}) k_\theta(\tilde{\mathbf{x}}_k, \boldsymbol{\psi}_{j,k})\, d\tilde{\mathbf{x}}_k \\
&= \int_{a_k}^{b_k} \exp\left\{ -\frac{(\tilde{\mathbf{x}}_k - \mathbf{x}'_k)^2 + (\tilde{\mathbf{x}}_k - \boldsymbol{\psi}_{i,k})^2 + (\tilde{\mathbf{x}}_k - \boldsymbol{\psi}_{j,k})^2}{\theta} \right\} d\tilde{\mathbf{x}}_k \\
&= \sqrt{\frac{\pi\theta}{12}} \exp\left\{ \frac{2}{3\theta} \left(\boldsymbol{\psi}_{i,k}\mathbf{x}'_k + \boldsymbol{\psi}_{j,k}\mathbf{x}'_k + \boldsymbol{\psi}_{i,k}\boldsymbol{\psi}_{j,k} - \mathbf{x}'^2_k - \boldsymbol{\psi}^2_{i,k} - \boldsymbol{\psi}^2_{j,k} \right) \right\} \\
&\quad \times \left(\operatorname{erf}\left\{ \frac{\iota_k^{(u,j)} - 3a_k}{\sqrt{3\theta}} \right\} - \operatorname{erf}\left\{ \frac{\iota_k^{(u,j)} - 3b_k}{\sqrt{3\theta}} \right\} \right),
\end{aligned}
\tag{3.10}
$$

notating $\mathbf{x}'_k$ as the kth entry of the vector $\mathbf{x}'$ and $\iota_k^{(u,j)} = \mathbf{x}'_k + \boldsymbol{\psi}_{u,k} + \boldsymbol{\psi}_{j,k}$.

The best new local inducing point can be found by solving the following program:

$$
\boldsymbol{\psi}_{m+1} = \operatorname{argmin}_{\boldsymbol{\psi}_{m+1} \in \mathcal{X}} \text{wIMSE}_n^{(m+1)}(\boldsymbol{\psi}_{m+1}, \mathbf{x}').
$$

The $\text{wIMSE}_n^{(m+1)}(\boldsymbol{\psi}_{m+1}, \mathbf{x}')$ surface realized over choices $\boldsymbol{\psi}_{m+1} \in \mathcal{X}$, which I shall visualize momentarily in Section 3.3.2, may be multi-modal. However, it is not pathologically so like a global IMSE. Library-based numerical schemes (details in Section 3.5.1) work well when suitably initialized but perform even better when aided by derivative information. The kth

component of the gradient is given by

$$\begin{aligned}
&\frac{\partial \text{wIMSE}(\boldsymbol{\psi}_{m+1}, \mathcal{X}, \boldsymbol{\Psi}_m, \mathbf{X}_n, \theta, \mathbf{x}')}{\partial \boldsymbol{\psi}_{m+1,k}} \\
&= -\text{tr}\left\{\left(\frac{\partial \mathbf{K}_{m+1}^{-1}}{\partial \boldsymbol{\psi}_{m+1,k}} - \frac{\partial \mathbf{Q}_{m+1}^{-1(n)}}{\partial \boldsymbol{\psi}_{m+1,k}}\right)\mathbf{W}'_{m+1}\right\} - \text{tr}\left\{\left(\mathbf{K}_{m+1}^{-1} - \mathbf{Q}_{m+1}^{-1(n)}\right)\frac{\partial \mathbf{W}'_{m+1}}{\partial \boldsymbol{\psi}_{m+1,k}}\right\} \\
&= \text{tr}\Big\{\Big(\mathbf{K}_{m+1}^{-1}\frac{\partial \mathbf{K}_{m+1}}{\partial \boldsymbol{\psi}_{m+1,k}}\mathbf{K}_{m+1}^{-1} - \mathbf{Q}_{m+1}^{-1(n)}\frac{\partial \mathbf{Q}_{m+1}^{(n)}}{\partial \boldsymbol{\psi}_{m+1,k}}\mathbf{Q}_{m+1}^{-1(n)}\Big)\mathbf{W}'_{m+1}\Big\} \\
&\quad - \text{tr}\left\{\left(\mathbf{K}_{m+1}^{-1} - \mathbf{Q}_{m+1}^{-1(n)}\right)\frac{\partial \mathbf{W}'_{m+1}}{\partial \boldsymbol{\psi}_{m+1,k}}\right\}.
\end{aligned} \tag{3.11}$$

In the matrix $\frac{\partial \mathbf{W}'_{m+1}}{\partial \boldsymbol{\psi}_{m+1,k}}$, all entries are zero except the row/column that corresponds to the row of $\boldsymbol{\Psi}_{m+1}$ that contains $\boldsymbol{\psi}_{m+1}$, which I place in the last $m+1^{\text{st}}$ row. The nonzero entries in $\frac{\partial \mathbf{W}'_{m+1}}{\partial \boldsymbol{\psi}_{m+1,k}}$, can be re-expressed as

$$\frac{\partial w'_{m+1}(\boldsymbol{\psi}_i, \boldsymbol{\psi}_{m+1})}{\partial \boldsymbol{\psi}_{m+1,k}} = \frac{\partial w_{m+1}^{'(i,m+1)}}{\partial \boldsymbol{\psi}_{m+1,k}} \prod_{k'=1,k'\neq k}^{d} w_{m+1,k'}^{'(i,m+1)},$$

where

$$\begin{aligned}
\frac{\partial w_{m+1}^{'(i,m+1)}}{\partial \boldsymbol{\psi}_{m+1,k}} &= \sqrt{\frac{\pi\theta}{12}}\exp\left\{\frac{2}{3\theta}\left(\boldsymbol{\psi}_{i,k}\mathbf{x}'_k + \boldsymbol{\psi}_{m+1,k}\mathbf{x}'_k + \boldsymbol{\psi}_{i,k}\boldsymbol{\psi}_{m+1,k} - \mathbf{x}'^2_k - \boldsymbol{\psi}_{i,k}^2 - \boldsymbol{\psi}_{m+1,k}^2\right)\right\} \\
&\times \left[\frac{2}{3\theta}(\mathbf{x}'_k - 2\boldsymbol{\psi}_{m+1,k} - \boldsymbol{\psi}_{i,k}) \times \left(\text{erf}\left\{\frac{\iota_k^{(i,m+1)} - 3a_k}{\sqrt{3\theta}}\right\} - \text{erf}\left\{\frac{\iota_k^{(i,m+1)} - 3b_k}{\sqrt{3\theta}}\right\}\right)\right. \\
&\left. + \frac{2}{\sqrt{3\pi\theta}}\left(\exp\Big\{-\frac{(\iota_k^{(i,m+1)} - 3a_k)^2}{3\theta}\Big\} - \exp\Big\{-\frac{(\iota_k^{(i,m+1)} - 3b_k)^2}{3\theta}\Big\}\right)\right].
\end{aligned} \tag{3.12}$$

Extensions to other kernel structures, such as Matérn (Stein, 2012), yield similar closed forms (i.e., further extending Binois et al., 2018b).

Fast matrix updates

Expressions for wIMSE and derivative (3.9–3.11) leverage the same Woodbury identities used earlier in equations (2.10, 3.5–3.6). Working with $\mathbf{K}_{m+1}$ and $\mathbf{Q}_{m+1}^{(n)}$ is cubic in m, yet even that is overkill. Thrifty evaluation of equations (3.9–3.11) lies in construction of $\mathbf{Q}_{m+1}^{(n)}$ which is equivalent to $\mathbf{K}_{m+1} + \mathbf{k}_{n,m+1}^\top \Omega_n^{-1(m+1)} \mathbf{k}_{n,m+1}$. Evaluating $\mathbf{k}_{n,m+1}^\top \Omega_n^{-1(m+1)} \mathbf{k}_{n,m+1}$ requires $2n-1$ products for each of $(m+1)^2$ entries, incurring costs in $\mathcal{O}(m^2 n)$ flops. Assuming $n \gg m$, this dominates the $\mathcal{O}(m^3)$ cost of decomposition.

More time can be saved through partitioned inverse (Barnett, 1979) sequential updates to $\mathbf{K}_{m+1}^{-1}$ after the new $\boldsymbol{\psi}_{m+1}$ is chosen, porting LAGPs frugal updates to the LIGP context. Writing $\mathbf{K}_{m+1}$ as an m-submatrix with new $m+1^{\text{st}}$ column gives

$$\begin{aligned} \mathbf{K}_{m+1} &= \begin{bmatrix} \mathbf{K}_m & \mathbf{k}_m(\boldsymbol{\psi}_{m+1}) \\ \mathbf{k}_m(\boldsymbol{\psi}_{m+1})^\top & k_\theta(\boldsymbol{\psi}_{m+1}, \boldsymbol{\psi}_{m+1}) \end{bmatrix} \quad \text{so that} \\ \mathbf{K}_{m+1}^{-1} &= \begin{bmatrix} \mathbf{K}_m^{-1} + \rho \boldsymbol{\Gamma}\boldsymbol{\Gamma}^\top & \boldsymbol{\Gamma} \\ \boldsymbol{\Gamma}^\top & \rho^{-1} \end{bmatrix}, \end{aligned} \tag{3.13}$$

using $\rho = k_\theta(\boldsymbol{\psi}_{m+1}, \boldsymbol{\psi}_{m+1}) - \mathbf{k}_m^\top(\boldsymbol{\psi}_{m+1}) \mathbf{K}_m^{-1} \mathbf{k}_m(\boldsymbol{\psi}_{m+1})$ and m-length column vector $\boldsymbol{\Gamma} = -\rho^{-1} \mathbf{K}_m^{-1} \mathbf{k}_m(\boldsymbol{\psi}_{m+1})$. Updating $\mathbf{K}_{m+1}^{-1}$ requires calculation of ρ, $\boldsymbol{\Gamma}$, and $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^\top$, each of which is in $\mathcal{O}(m^2)$. Thus I reduce the computational complexity of $\mathbf{K}_{m+1}^{-1}$ from $\mathcal{O}(m^3)$ to $\mathcal{O}(m^2)$. Similar partitioning provides sequential updates to $\Omega_n^{(m+1)}$, a diagonal matrix:

$$\begin{aligned} \Omega_n^{(m+1)} &= \text{Diag}\big(\mathbf{K}_n + \epsilon_K \mathbb{I}_n - \mathbf{k}_{n,m+1} \mathbf{K}_{m+1}^{-1} \mathbf{k}_{n,m+1}^\top\big) \\ &= \Omega_n^{(m)} - \rho^{-1} \text{Diag}\left\{\zeta\zeta^\top\right\}, \end{aligned} \tag{3.14}$$

where $\zeta = \mathbf{k}_{nm} \mathbf{K}_m^{-1} \mathbf{k}_m(\boldsymbol{\psi}_{m+1}) - \mathbf{k}_n(\boldsymbol{\psi}_{m+1})$. Updates of $\Omega_n^{(m+1)}$ without partitioning, driven

by matrix–vector product(s) $\mathbf{k}_{n,m+1}\mathbf{K}_{m+1}^{-1}\mathbf{k}_{n,m+1}^{\top}$ involve m^2n flops. Using (3.14) reduces the cost of the updates to $\mathcal{O}(mn)$.

Unlike in equation (3.13), $\mathbf{Q}_m^{(n)}$ cannot be trivially augmented to construct $\mathbf{Q}_{m+1}^{(n)}$ due to the presence of $\Omega_n^{(m)}$ which is also embedded in $\mathbf{Q}_m^{(n)}$. Yet there are some time savings to be found in the partitioned inverses

$$\begin{aligned} \mathbf{Q}_{m+1}^{(n)} &= \begin{bmatrix} \mathbf{Q}_{m+}^{(n)} & \boldsymbol{\gamma}(\boldsymbol{\psi}_{m+1}) \\ \boldsymbol{\gamma}(\boldsymbol{\psi}_{m+1})^{\top} & \chi(\boldsymbol{\psi}_{m+1}) \end{bmatrix} \quad \text{and} \\ \mathbf{Q}_{m+1}^{-1(n)} &= \begin{bmatrix} \mathbf{Q}_{m+}^{-1(n)} + \upsilon\boldsymbol{\xi}\boldsymbol{\xi}^{\top} & \boldsymbol{\xi} \\ \boldsymbol{\xi}^{\top} & \upsilon^{-1} \end{bmatrix}, \end{aligned} \tag{3.15}$$

with $\mathbf{Q}_{m+}^{(n)} = \mathbf{K}_m + \mathbf{k}_{nm}^{\top}\Omega_n^{(m+1)-1}\mathbf{k}_{nm}$ built via updated values of $\Omega_n^{(m+1)}$, $\boldsymbol{\gamma}(\boldsymbol{\psi}_{m+1}) = \mathbf{k}_m(\boldsymbol{\psi}_{m+1}) + \mathbf{k}_{nm}^{\top}\Omega_n^{-1(m+1)}\mathbf{k}_n(\boldsymbol{\psi}_{m+1})$, $\chi(\boldsymbol{\psi}_{m+1}) = k_\theta(\boldsymbol{\psi}_{m+1}, \boldsymbol{\psi}_{m+1}) + k_n(\boldsymbol{\psi}_{m+1})^{\top}\Omega_n^{-1(m+1)}k_n(\boldsymbol{\psi}_{m+1})$, $\upsilon = \chi(\boldsymbol{\psi}_{m+1}) - \boldsymbol{\gamma}(\boldsymbol{\psi}_{m+1})^{\top}\mathbf{Q}_{m+}^{-1(n)}\boldsymbol{\gamma}(\boldsymbol{\psi}_{m+1})$ and $\xi = -\upsilon^{-1}\mathbf{Q}_{m+}^{-1(n)}\boldsymbol{\gamma}(\boldsymbol{\psi}_{m+1})$. Similar to $\mathbf{Q}_m^{(n)}$, calculating $\mathbf{Q}_{m+}^{(n)}$ requires in flops in $\mathcal{O}(m^2n)$. Consequently the entire scheme can be managed in $\mathcal{O}(m^2n)$.

3.3.2 Illustrations of Greedy Inducing Point Search

Greedily optimizing wIMSE to place local inducing points around neighborhood $\mathbf{X}_n(\mathbf{x}')$ results in $\boldsymbol{\Psi}_m(\mathbf{x}')$ with (approximately) minimal predictive variance nearby $\mathbf{x}'$, so naturally they concentrate in that locale. To explore inducing point optimization with wIMSE, I use a toy 2d test problem known as Herbie's tooth (Lee et al., 2011). This function is attractive due to its low dimensionality but complex non-stationary surface littered with local minima. The function is defined by $z(x_1, x_2) = -w(x_1)w(x_2)$ where $w(x) = \exp\{-(x-1)^2\} + \exp\{-0.8(x+1)^2\} - 0.05\sin(8(x+0.1))$ and $x_1, x_2 \in [-2, 2]$.

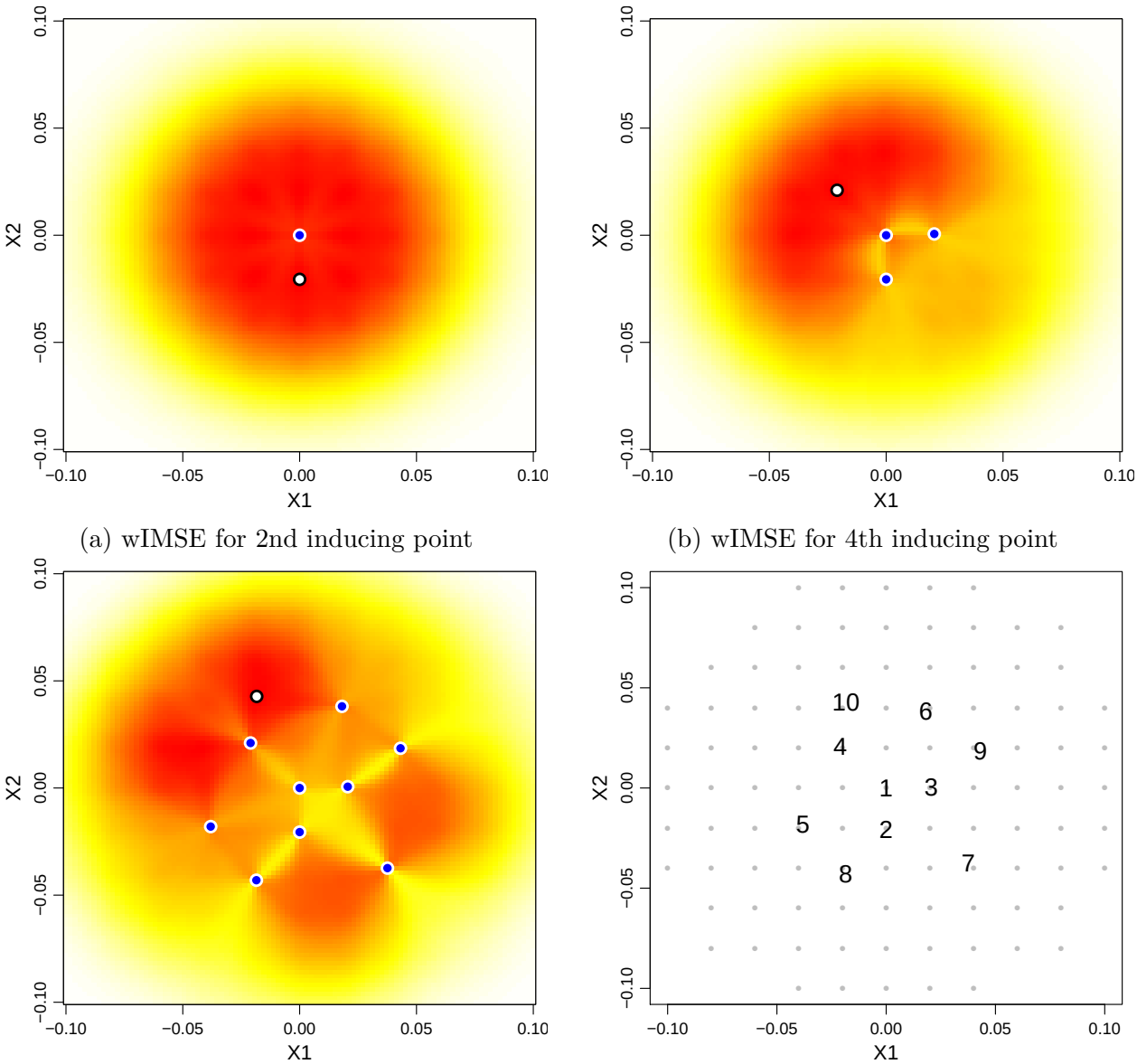
Figure 3.4 shows the evolution of wIMSE-based acquisition for $\mathbf{x}'$ placed at the origin for Herbie's tooth ($N = 40\text{K}$, $n = 100$). Panels (a–c) show existing $\boldsymbol{\Psi}_m(\mathbf{x}')$ in blue overlayed on the wIMSE surface used to select $\boldsymbol{\psi}_{m+1}$. Optimal $\boldsymbol{\psi}_{m+1}$, i.e., the wIMSE global minimum, are represented by white-filled circles. Unlike global VFE likelihood, ALC, and IMSE surfaces (Figure 3.2), the local wIMSE surface does not appear to be as affected by placement of the training points $\mathbf{X}_N$, or local neighborhood $\mathbf{X}_n(\mathbf{x}') \subset \mathbf{X}_N$, shown as dots in panel (d). Local minima still exist as more inducing points are introduced. Yet the wIMSE surface is much smoother and well-behaved, making optimization easier.

The first selection, $\boldsymbol{\psi}_1(\mathbf{x}')$, often lies very close to $\mathbf{x}'$. When $\mathbf{x}'$ is near the boundary of the input space, where wIMSE would be asymmetric, the first inducing point selection may "pull away" somewhat from $\mathbf{x}'$ toward to middle of the space. But when symmetry is high, as it is at the origin for the illustration in Figure 3.4, it is hard to distinguish between $\boldsymbol{\psi}_1$ and $\mathbf{x}'$ up to numerical error. I find it convenient to simply begin optimizing at iteration two, with $\boldsymbol{\psi}_1 = \mathbf{x}'$.

Algorithm 2 Inducing Point wIMSE Design

```
procedure IP.wIMSE(m, n, x', X, 𝒳)
    X_n ← NN(x', X, n)                                  ▷ Find n nearest neighbors to x'
    θ^(0) ← quantile(0.1, dist(X_n))                    ▷ Reasonable local lengthscale
    ψ_1 ← x';                                           ▷ Place first inducing point
    for i = 2, ..., m do                                ▷ Greedy wIMSE to find the rest
        ψ_i ← argmin_{ψ_i ∈ 𝒳} wIMSE_n^(i)(ψ_i, x')     ▷ Implicit dependence on θ^(0)
    end for                                             ▷ Implicit updates of local induced GP
    return Ψ_m(x') = {ψ_i}_{i=1}^m and X_n(x') = X_n
end procedure
```

For concreteness, steps for this greedy wIMSE inducing point search are outlined in Algorithm 2. After building the local neighborhood $\mathbf{X}_n(\mathbf{x}')$, initialization is completed by choosing $\boldsymbol{\psi}_1 \leftarrow \mathbf{x}'$ and local lengthscale $\theta^{(0)}$. Here I set $\theta^{(0)}$ based on quantiles of squared distances in $\mathbf{X}_n(\mathbf{x}')$, though other settings are considered later. After greedy selection over



(a) wIMSE for 2nd inducing point

(b) wIMSE for 4th inducing point

(c) wIMSE for 10th inducing point

(d) Local neighborhood and inducing points

Figure 3.4: wIMSE surfaces (a–c), red/lower yellow/higher, used to optimize the 2nd, 4th, and 10th inducing points: existing in blue; new selection in white. Predictive location $\mathbf{x}'$ is at the origin, which is where $\boldsymbol{\psi}_1$ is placed. Panel (d) summarizes the neighborhood $\mathbf{X}_n(\mathbf{x}')$ as gray dots and local inducing points $\boldsymbol{\Psi}_m(\mathbf{x}')$ in number order.

$i = 1, \ldots, m$, intermixed with updates to the locally induced GP structure as outlined in Section 3.3.1, the procedure returns an $m \times d$ matrix comprised of the selected inducing points $\boldsymbol{\Psi}_m(\mathbf{x}')$ alongside an $n \times d$ matrix defining the local neighborhood $\mathbf{X}_n(\mathbf{x}')$.

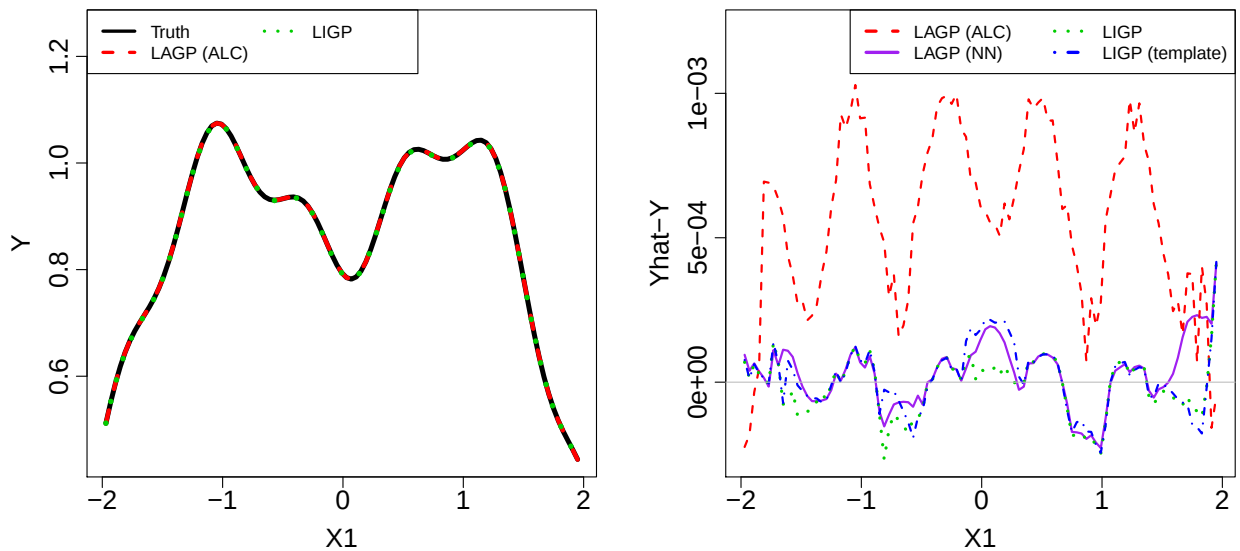


Figure 3.5: *Left*: approximate GP fits' mean prediction and truth on a slice of Herbie's tooth at $x_2' = 0.6$. *Right*: errors relative to the truth on the approximate GP fits for the same slice of Herbie's tooth.

The left panel of Figure 3.5 shows the predictions for a grid of $\mathbf{x}'$ settings arranged over a 1d slice of Herbie's tooth where $x_2' = 0.6$, including LAGP (via ALC with $n = 50$, defaults in `laGP`) and LIGP $(m, n) = (10, 100)$, with local subset and inducing point designs re-optimized at each predictive location. I allow LIGP a bigger neighborhood (n), with explanation in Section 3.4.3, but remind that this involves thriftier m-sized cubic decompositions. Observe that both LAGP (red-dashed) and LIGP (green-dotted) capture the bumpiness of the surface, completely overlaying the true out-of-sample response (black-solid).

Zooming in, the right panel of Figure 3.5 shows errors along the slice under these comparators and two new variations: LAGP via NN with $n = 100$ and LIGP with $(m, n) = (10, 100)$ via template (Section 3.4.1). Along most of the slice, LIGP's error follows a similar trend as LAGP (NN, $n = 100$), albeit with a bumpier line. This is not surprising given that both GP fits use the same neighborhood $\mathbf{X}_n(\mathbf{x}')$. LAGP (ALC) copes well with smaller $n = 50$ by filling $\mathbf{X}_n(\mathbf{x}')$ with a mix of NNs and satellites.[4] Averaging along that slice,

[4]For identical n, ALC bests NN (Gramacy and Apley, 2015), motivating increased n for NN here.

out-of-sample RMSE for LAGP (ALC) was 7.88×10^{-4}, versus 1.14×10^{-4} and 1.12×10^{-4} for LAGP (NN) and LIGP, respectively. Here, LIGP predicts slightly better than LAGP (NN), its most direct competitor, and noticeably better than LAGP (ALC). By reducing the computational burden of the optimization criteria (NN v. ALC) and matrix inversions (LIGP v. LAGP), I free up resources to increase n and thus accuracy.

Encouraging as these early LIGP results are, selecting novel $\mathbf{\Psi}_m(\mathbf{x}')$ for each $\mathbf{x}'$ is a substantial undertaking. LIGP required 3.32 seconds, on average, to greedily build $\mathbf{\Psi}_m(\mathbf{x}')$ using about 9 derivative-based iterates at each $\mathbf{x}'$. Once in hand, optimizing via likelihood using a local analog of equation (3.3) and predicting (3.6) based on $\mathbf{\Psi}_m(\mathbf{x}')$ and $\mathbf{X}_n(\mathbf{x}')$ is almost instantaneous, requiring 0.0062 seconds per prediction. LAGP (NN or ALC), which search discretely over subsets, lag a little behind at 0.0437 and 0.073 seconds, respectively.

3.4 Refinements to neighborhood composition

LIGP can be accelerated with little impact on predictive accuracy by applying a single inducing point design $\mathbf{\Psi}_m(\mathbf{x}')$ almost identically over all predictive locations $\mathbf{x}' \in \mathcal{X}$ of interest. Here, I explore the benefits of inducing point design templates built with wIMSE and thriftier space-filling strategies.

3.4.1 Inducing points template

Creating $\mathbf{\Psi}_m(\mathbf{x}')$ based on wIMSE for each $\mathbf{x}' \in \mathcal{X}$ is a chore that can cannibalize any benefit that might come with adopting an inducing point approximation in the first place. The highly structured nature of optimal wIMSE-based inducing points (Figure 3.4d) suggests such effort might be overkill. Perhaps the cost of a single, representative optimization could

be amortized over the expense of its application on a vast predictive grid. When re-purposed, through shifting or other transformation for new $\mathbf{x}'$, I refer to the original wIMSE design – which might be calculated at the middle of the input space – as a *template*.

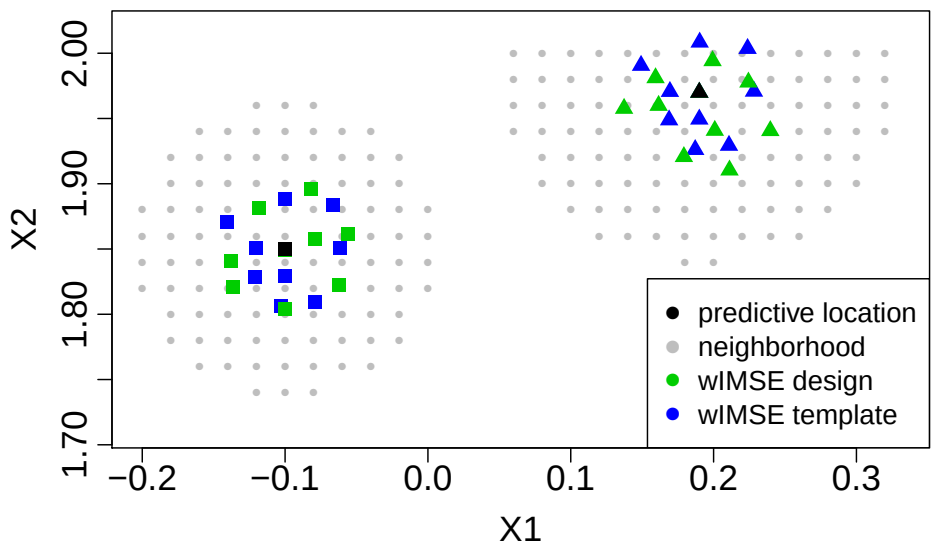


Figure 3.6: Local neighborhoods for two predictive locations $\mathbf{x}'$ at $(-0.1,\ 1.85)$ and $(0.19,\ 1.97)$. Gray dots are $n = 100$ neighborhoods $\mathbf{X}_n(\mathbf{x}')$. Green points are wIMSE optimal inducing points $\mathbf{\Psi}_m(\mathbf{x}')$; blue ones are displaced templates derived at the origin. The wIMSE template performs nearly the same space-filling effect as the locally optimized inducing points.

Figure 3.6 depicts the essence of the idea, comparing bespoke $\mathbf{\Psi}_m(\mathbf{x}')$ to re-shifted ones from a template in two variations. The setup is again Herbie's tooth in $[-2, 2]^2$ and the two predictive sites are $\mathbf{x}'^{(1)} = (-0.1, 1.85)$ and $\mathbf{x}'^{(2)} = (0.19, 1.97)$ whose $n = 100$ neighborhoods $\mathbf{X}_n(\mathbf{x}'^{(1)})$ and $\mathbf{X}_n(\mathbf{x}'^{(2)})$, shown as gray dots, reside completely in the interior and on the x_2 boundary, respectively. Blue points in the plot represent a wIMSE-based inducing point design – as optimized (Section 3.3) at the center of the design space and then – shifted to be centered at the $\mathbf{x}'$s. Compare these template-based local inducing points to corresponding optimal analogues in green. At both predictive locations, the pair of inducing point designs differ, yet both still space-fill the inner-neighborhood around $\mathbf{x}'$. A mild exception may be template-based $\mathbf{\Psi}_m(\mathbf{x}'^{(2)})$ with its two points outside of the design region, which would not

happen under an exhaustive re-optimization. Other differences between alternatives would otherwise appear to be cosmetic up to rotation/small perturbations as may stem from a myriad of benign causes: relationship of $\mathbf{x}'$ to its local neighborhood $\mathbf{X}_n(\mathbf{x}')$, convergence and global scope in greedy optimization, etc.

Looking back at the right panel of Figure 3.5, observe how prediction errors based on templates (blue dashed line) compare with locally wIMSE-optimized inducing points (green dotted line) along the slice. Both LIGP variations seem to underestimate the response compared to LAGP (NN), but the template methods give nearly as accurate predictions as LIGP with locally wIMSE-optimized inducing points. Transferring a template captures most of the variability between local wIMSE designs, even at the boundaries. The template is also much faster. It took a total of 328.82 seconds to fit separate $\mathbf{\Psi}_m(\mathbf{x}')$ and predict at the 99 $\mathbf{x}'$ locations depicted in the slice. Using a template instead takes 3.82 seconds, a near two orders of magnitude improvement.

Algorithm 3 Building and Displacing Inducing Point Templates

```
1: x̌ ← median(X)                                   ▷ Set x̌ to the center of the data
2: Ψ_m ← IP.wIMSE(m, n, x̌, X, 𝒳)                    ▷ Use Alg. 2 on x̌
3: Ψ*_m ← Ψ_m − x̌                                   ▷ Center template at the origin
4: procedure IP.Template(n, x′, X, Ψ′_m)
5:     X_n ← NN(x′, X, n)
6:     Ψ_m ← Ψ′_m + x′                              ▷ Simple displacement
7:     return Ψ_m(x′) = {ψ_i}_{i=1}^m and X_n(x′) = X_n
8: end procedure
```

Algorithm 3 provides pseudocode for this template scheme, clarifying how a single wIMSE-based local inducing point design $\mathbf{\Psi}_m$ is displaced for each $\mathbf{x}'$. It is worth remarking that the scheme makes a tacit presumption that the full design structure, $\mathbf{X}_N$, is somewhat homogeneous: similar near the middle of the input space, $\check{\mathbf{x}}$, as near where it will be applied, i.e., for many disparate $\mathbf{x}' \in \mathcal{X}$. I do not doubt it would be possible to engineer test problems, and/or non-space-filling designs $\mathbf{X}_N$, that would thwart this scheme, yet I find it works well

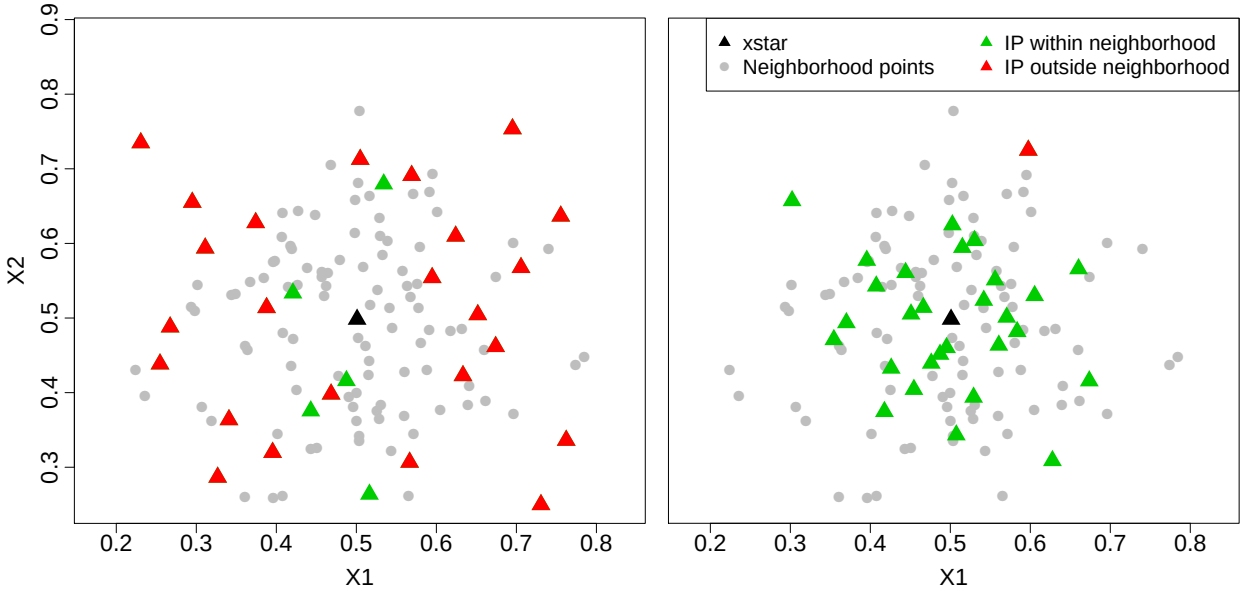
in most cases.

3.4.2 Space-filling templates

My template-scheme leverages the neighborhood-focused space-filling nature of inducing points, beyond say $\boldsymbol{\psi}_1 \approx \mathbf{x}'$. Space-fillingness is a cornerstone of (global) computer experiment design. Numerous schemes exist, such as Latin hypercube samples (LHSs McKay et al., 2000) or maximin designs (Johnson et al., 1990), etc., and hybrids thereof (Morris and Mitchell, 1995). These work well and often require less computation than model-based alternatives such as IMSE. If such space-filling designs (SFDs) could be re-tooled to "focus" on particular parts of the input space – say in the neighborhood of $\mathbf{x}'$ – I might be able to avoid an expensive greedy wIMSE optimization all together. SFDs might be able to mimic the behavior of a wIMSE template scheme at almost no cost at all.

SFDs are usually constructed in a unit hypercube. Re-centering such a template to $\mathbf{x}'$ is trivial, but re-scaling so that it lies within $\mathbf{X}_n(\mathbf{x}')$ and resembles $\boldsymbol{\Psi}_m(\mathbf{x}')$ is more challenging. One way is to derive a second, local rectangle as a means of defining a linear mapping between scales. A thrifty strategy is to use the bounds of the neighborhood $\mathbf{X}_n(\mathbf{x}')$. But the shape of $\mathbf{X}_n(\mathbf{x}')$ is roughly spherical, being comprised of Euclidean distance-based NNs. Thus the rectangular SFD will cover regions outside of the hypersphere, potentially placing some inducing points outside the neighborhood. In low input dimension, say $d \leq 2$, this is no big deal, because the circumscription is relatively tight. But when $d = 8$, say, circumscription is poor.

Figure 3.7a shows a 2d projection of an 8d local neighborhood for the borehole problem, described in Section 3.5.2. Here, the volume of the convex hull of the neighborhood $\mathbf{X}_n(\mathbf{x}')$ is less than one sixtieth of the size of the rectangle circumscribing its bounds in



(a) LHS rectangular template (b) LHS Φ template

Figure 3.7: SFD template schemes (triangles) in 2d projections relative to local neighborhood (gray dots): (a) rectangular re-scaled LHS template (triangles) in relation to a local neighborhood (gray dots); (b) qNorm LHS template. Green triangles indicate $\mathbf{\Psi}_m(\mathbf{x}')$ within the neighborhood $\mathbf{X}_n(\mathbf{x}')$ in all coordinates; red outside.

the coordinate axis directions. Consequently many of the template re-scaled local inducing points $\mathbf{\Psi}_m(\mathbf{x}')$, indicated as triangles, lie outside the neighborhood (red) in at least one of the eight coordinates. Of the $m = 30$ local inducing points calculated for that figure, one of which is automatically at $\mathbf{x}'$, only six rectangular re-scaled LHS template points lie within the neighborhood.

As remedy, I propose a nonlinear mapping that warps the SFD to lie inside the neighborhood with high probability. In particular, I scale the SFD based on an inverse Gaussian CDF (Φ^{-1}), applied separately to each of the d input coordinates. Algorithm 4 outlines steps toward generating an inducing point design $\mathbf{\Psi}_m(\mathbf{x}')$ based on a SFD $\hat{\mathbf{X}}$ of size $m-1$, i.e., beyond choosing $\boldsymbol{\psi}_1 = \mathbf{x}'$. Φ^{-1} calculations for each dimension $k = 1, \ldots, d$ involve $\mu = \mathbf{x}'_k$ and variance $\theta^{(0)}$. This is the same $\theta^{(0)}$ as in Algorithm 2 for greedy wIMSE optimization,

except here I demonstrate a more absolute default choice. This Φ^{-1} transformation yields higher density near $\mathbf{x}'$ and much lower density outside of the neighborhood's hypersphere. Observe in Figure 3.7b how this warping drastically reduces the number of template points outside of the neighborhood.

Algorithm 4 Inverse Gaussian CDF Space-Filling Template

```
procedure IP.qNorm(m, n, x', X)
    X_n ← NN(x', X, n)                                  ▷ Find n nearest neighbors to x'
    θ^(0) ← (1/3 max_k |X_{n,k}(x') − x'_k|)^2          ▷ Reasonable local lengthscale
    X̂ ← SFD[0,1]^d with m − 1 points                    ▷ Could be moved outside
    for k = 1, ..., d do                                ▷ Warp each input coordinate
        x̆_k ← Φ^{-1}(x̂_k; μ = x'_k, σ^2 = θ^(0))        ▷ Inverse Gaussian CDF with μ, σ^2
    end for
    Ψ_m ← rowbind(x', X̆)                                ▷ Add x' as inducing point
    return Ψ_m(x') = Ψ_m and X_n(x') = X_n
end procedure
```

Pseudocode in Algorithm 4 conveys bespoke SFD within each application of the subroutine, yielding new $\hat{\mathbf{X}}$ in each call. As with the wIMSE template in Algorithm 3, this can be moved outside the subroutine to fix a single SFD, which might be important if the SFD is expensive to compute. I prefer LHSs for my SFDs because they are easy/instantaneous via libraries such as `lhs` (Carnell, 2019) on CRAN. Hybrids such as maximin–LHS are also straightforward (also with `lhs`), which can avoid some pathologies inherent in random LHS design. Ordinary maximin can be problematic under Φ^{-1} because that criteria places points on the bounding hypercube, which would warp to $\pm\infty$ without intervention, and because evaluating and optimizing that criteria are slow. Uniformly random design may be preferred when local lengthscales are difficult to estimate (Zhang et al., 2019).

Section 3.3.2 offered comparison between run time and predictive accuracy for LIGP, using wIMSE to build unique inducing point designs, to that of LAGP on a slice of Herbie's tooth. Now consider new template comparators: hyperrectangular SFD, LIGP (cHR), and

Φ^{-1}-scaled SFD, LIGP (qNorm). While it took 3.32 seconds on average to build wIMSE-based designs, scaling an SFD to circumscribe the neighborhood (cHR) or applying Φ^{-1} (qNorm) only takes 0.01 seconds on average. Both of these SFD template schemes produce an RMSE that is essentially the same (1.8×10^{-4}) as applying the wIMSE template scheme.

The borehole problem uses larger $(m, n) = (80, 150)$ settings due to the higher input dimension (discussed in Section 3.4.3). It takes 141 seconds to build a wIMSE-based inducing point template of size m, while it only takes 0.034 seconds to build a SFD-scaled template. SFD and wIMSE templates produce LIGPs with similar RMSEs, discussed in Section 3.5.2.

3.4.3 Determining neighborhood size

Little attention is paid in the literature to the choosing the number of (global) inducing points (Azzimonti et al., 2016; Seeger et al., 2003; Titsias, 2009) relative to problem size (N, d), except on computational grounds – smaller M is better. The same is true for local neighborhood size n in LAGP. Although there is evidence that the `laGP` default of $n = 50$ is too small (Gramacy, 2016), especially with larger input dimension d, cubically growing expense in n limits the efficacy of larger n in practice. With local inducing points this is mitigated through cubic-in-m proxies, allowing larger local neighborhoods, thus implying more latitude to explore/choose good (m, n) combinations.

Toward that end, I considered a coarse grid of (m, n) and predictive RMSEs on Herbie's tooth ($d = 2$) and borehole ($d = 8$) toy problems. Setup details are identical to descriptions in Sections 3.3.2 and 3.5.2, respectively, and I used the qNorm (Φ^{-1}) template throughout. An LHS testing set of size $N' = 1000$ was used to generate the response surfaces of RMSEs reported in Figure 3.8. These are shown in log space for a more visually appealing color scheme, and were obtained after GP smoothing to remove any artifacts from random testing.

Grid elements where $m > n$ were omitted from the simulation on the grounds that there are no run-time benefits to those choices.

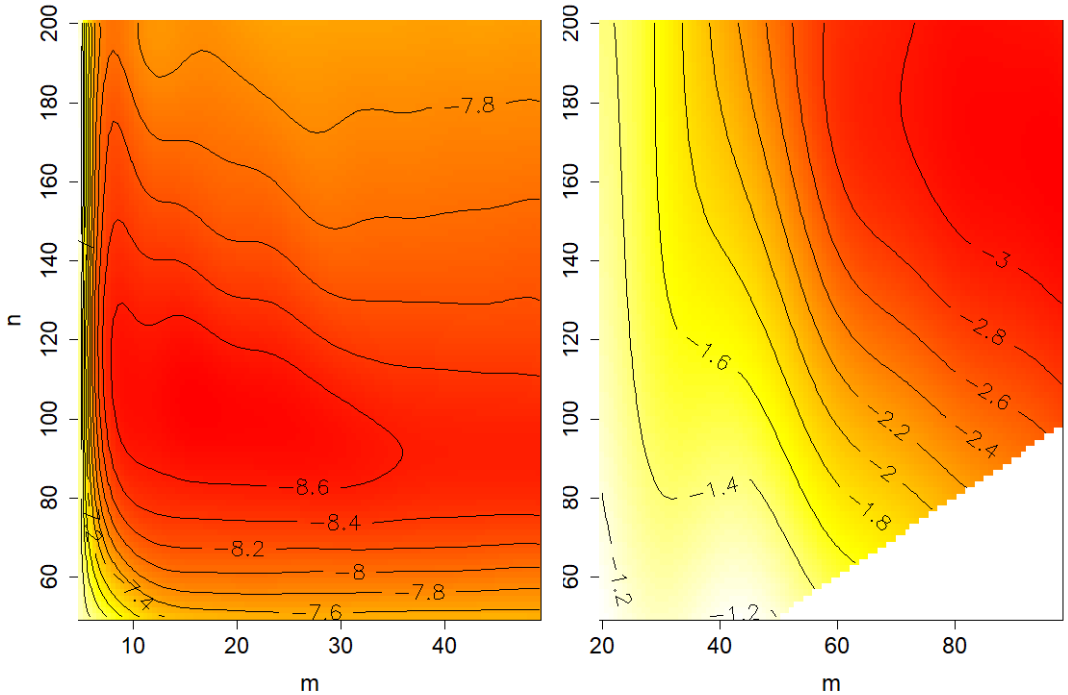


Figure 3.8: log(RMSE) over inducing points m and neighborhood n: Herbie's tooth (left) and borehole (right).

Observe that both surfaces are fairly flat across a wide swath of m, excepting quick ascent (decrease in accuracy) for smaller numbers of inducing points in the left panel. The situation is similar for n. Best settings are apparently input-dimension dependent. Numbers of inducing points as low as $m = 10$ seems sufficient in 2d (top panel), whereas $m = 80$ is needed in 8d. For borehole, it appears that larger neighborhoods n are better, perhaps because the response surface is very smooth and the likelihood prefers long lengthscales (Gramacy, 2016). A setting like $n = 150$ seems to offer good results without being too large. The situation is different for Herbie's tooth. Here, larger n has deleterious effects. Its non-stationary nature demands reactivity which is proffered by smaller local neighborhood. A setting of $n = 100$ looks good.

These are just two problems, and it is clearly not reasonable to grid-out (m, n) space for all future applications. But nevertheless, I have found that these rules of thumb port well to my empirical work in Section 3.5. The satdrag example ($d = 8$) and classic $d = 21$ benchmark (in Section 3.5) work well with the settings found for borehole, for example. Some ideas for automating the choice of (m, n) are discussed in Section 3.6.

3.5 Computation and benchmarking

Here I provide implementation details followed by in-depth comparison of LIGP and various template schemes, to LAGP on a swath of synthetic and real computer simulation experiments. My metrics for benchmarking are out-of-sample RMSE and computation time. All analysis was performed on an eight-core hyperthreaded Intel i9-9900K CPU at 3.60 GHz.

3.5.1 Implementation details

R code (R Core Team, 2021) supporting my methodological contribution, and all examples, may be found on my Git repository.

https://bitbucket.org/gramacylab/lagp/src/master/R/inducing/

Some noteworthy aspects of that implementation include the following. Unlike `laGP`, which is coded in C with `OpenMP` for symmetric multiprocessing parallelization (R serving only as wrapper), my LIGP implementation is pure R. Nevertheless, my template schemes are competitive, time-wise, and sometimes notably faster.

I privilege an isotropic Gaussian kernel formulation with scalar lengthscale θ for local modeling, although there is no reason other forms, such as Matérn (Stein, 2012), could not

be entertained so long as the structure is differentiable with respect to inducing points $\mathbf{\Psi}_m$. To improve numerical conditioning of matrices $\mathbf{K}_m$ and $\mathbf{Q}_m^{(n)}$ for stable inversion, I augment their diagonals with $\epsilon_K = 10^{-6}$ and $\epsilon_Q = 10^{-5}$ jitter (Neal, 1998), respectively. While both are theoretically decomposable, I find that $\mathbf{Q}_m^{(n)}$ is more sensitive to conditioning issues, thus requiring larger ϵ_Q. In the context of LAGP, it has been shown that separable local formulations do not much improve predictive performance, especially after first applying a global pre-scaling of inputs (Sun et al., 2019a). Such stretching and compressing of inputs,[5] has recently become popular as a means of boosting predictive performance of approximate GP methods (e.g., Katzfuss et al., 2020). When pre-scaling in my exercises to ensure apples-with-apples comparisons to benchmarks I divide by square-root separable global lengthscales obtained from a GP's fit to random size-1000 data subsets. See Gramacy (2020), Section 9.3.4, for details. The time required is not included in my summaries.

Building of wIMSE inducing point designs $\mathbf{\Psi}_m(\mathbf{x}')$ and templates $\mathbf{\Psi}_m(\check{\mathbf{x}})$, generically $\mathbf{\Psi}_m$ below, follows Algorithm 3 with m and n appropriate to the input dimension d (Section 3.4.3), provided momentarily with my particular exercises. For initial local lengthscale $\theta^{(0)}$, I have had success with a number of heuristics which often lead to similar values/performance for LIGP methods in my exercises. Gramacy (2016) suggests the 10% quantile of squared pairwise distances between the neighborhood points $\mathbf{X}_n$.[6] See Algorithm 2. A downside is that this is quadratic in n. A more absolute/direct $\mathcal{O}(n)$ approach matches $\theta^{(0)} = \sigma^2$, where 3σ approximates the 99% quantile of a Gaussian fit, to the margins of $\mathbf{X}_n$. Algorithm 4 exemplifies this choice for contrast, although I see these as interchangeable. Each $\boldsymbol{\psi}_{m+1}$ augmenting $\mathbf{\Psi}_m$ optimizing wIMSE is found via a 20-point multi-start L-BFGS-B (Byrd et al., 1995) scheme (using `optim` in R) peppered within the bounding box surrounding the neighborhood $\mathbf{X}_n$ to a tolerance of 0.01. Templates derived from space-filling designs

[5] A characterization attributed to Derek Bingham predating any published account, to my knowledge.
[6] In `laGP`, the function providing $\theta^{(0)}$ in this way is `darg`.

(Section 3.4.2) originate from $m-1$ point LHSs through the hyperrectangle enclosing $\mathbf{X}_n(\check{\mathbf{x}})$, and then augmented with $\check{\mathbf{x}}$ as the mth inducing point.

Regardless of inducing point/template construction, machinery behind LIGP-based prediction is identical. Algorithm 1 outlines the steps to construct local neighborhoods and predict at each of a set of N' prediction locations $\mathbf{X}'$ given training data $\{\mathbf{X}_N, \mathbf{Y}_N\}$, neighborhood size n, and number of inducing points m. Each location $\mathbf{x}_i$, for $i = 1, \ldots, N'$ could proceed in parallel. In my implementation I use 16 threads.[7] The pseudocode attempts to be agnostic about the inducing point scheme by simply writing IP(...). Any of Algorithms 2–4 can be used here. To estimate scale and lengthscale I used equations (3.4–3.5) through simple substitutions of (m, n) for the local neighborhoods of $\mathbf{x}'$. I rely on `optim` in R to minimize the negative log-likelihood to estimate local $\hat{\theta}(\mathbf{x}')$'s. Finally, the predictive mean and variance for $\mathbf{x}'$ are extracted via equation (3.6).

3.5.2 Borehole

Previewed in Section 3.4.2, the borehole function (Worley, 1987) is a classic example in computer experiments literature. Outputs may be derived in closed form as

$$y = \frac{2\pi T_u[H_u - H_l]}{\log\left(\frac{r}{r_w}\right)\left[1 + \frac{2LT_u}{\log(r/r_w)r_w^2 K_w} + \frac{T_u}{T_l}\right]},$$

[7] That is, two per hyperthreaded core.

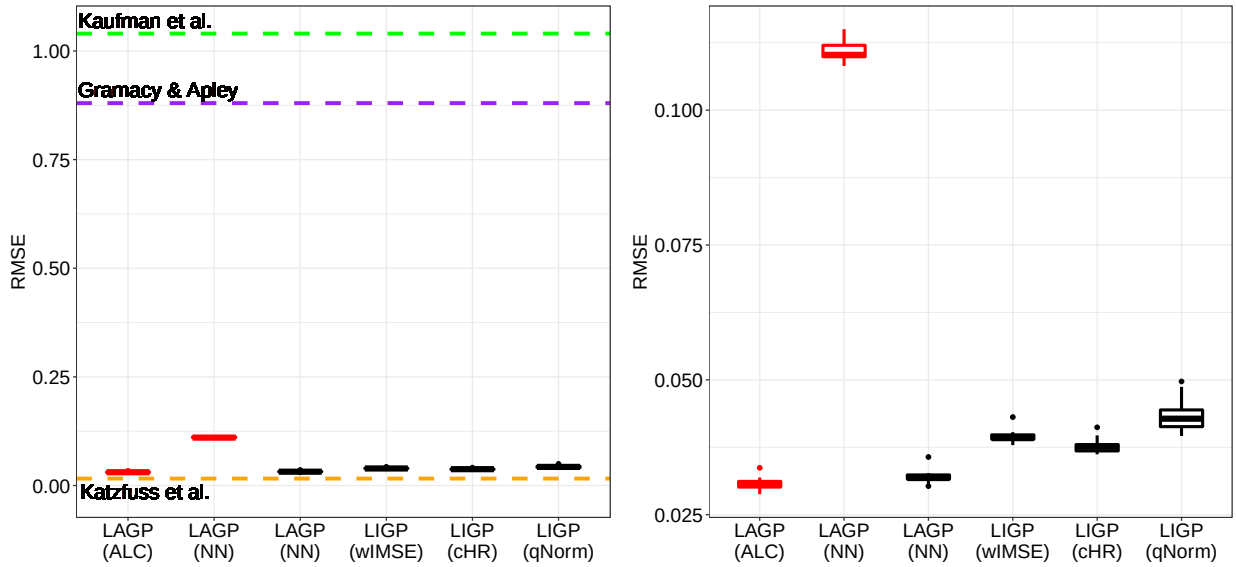
via inputs in the eight-dimensional rectangle:

$$r_w \in [0.05, 0.15] \qquad r \in [100, 5000]$$
$$T_u \in [63070, 115600] \qquad T_l \in [63.1, 116]$$
$$H_u \in [990, 1100] \qquad H_l \in [700, 820]$$
$$L \in [1120, 1680] \qquad K_w \in [9855, 12045].$$

For training I use LHSs of size $N = 100000$, recoding natural inputs to the unit 8-cube followed by pre-scaling via a global separable $\hat{\theta}$ as explained in Section 3.5.1. I use $(m, n) = (80, 150)$ for all LIGP fits (Section 3.4.3). For a fair comparison, I entertain $n = 150$ for LAGP (NN) as well as the default of $n = 50$ for NN and ALC-based LAGP comparators. Figure 3.9 summarizes RMSEs obtained over thirty MC instances with novel training and $N' = 10000$ sized LHS testing sets.

Mirroring other studies (e.g., Sun et al., 2019a), local approximation is key to using a vast training data set to get good predictions. LAGP performs better with a neighborhood of $n = 50$ selected using ALC versus even larger neighborhoods ($n = 150$) using NN. Given the smoothness of the borehole surface, the addition of "satellite" points provided by ALC gives an accuracy boost over pure NN of similar size. I believe the same to be true of LIGP (cHR). Any inducing points lying outside the neighborhood act as "satellites" in this context. This is backed up by comparable RMSE results. The added flexibility of inducing points (LIGP) over discrete subsets (LAGP) may be limited by the highly smooth borehole dynamics.

Timings are provided at the bottom of Figure 3.9, with LIGP LHS templates being fastest among the most competitive alternatives, accuracy-wise. Interestingly, the cHR template is even better at prediction than the optimized wIMSE one, obtained at great computational expense (3.06 minutes). Compared to LAGP (NN) with $n = 150$, accuracy is



Method	LAGP	LAGP	LAGP	LIGP	LIGP	LIGP
Variation	ALC	NN	NN	wIMSE	cHR	qNorm
n	50	50	150	150	150	150
Time	0.97	0.08	1.27	3.06	0.73	0.73

Figure 3.9: *Top-left:* accuracy over 30 MC repetitions with lines showing other published works' results: (Kaufman et al., 2011, green), (Gramacy and Apley, 2015, purple), (Katzfuss et al., 2020, orange). *Top-right:* zoomed in version focusing on the best LI/LAGP methods. The color of the boxplot outline, red and black, correspond to the sizes of the neighborhoods ($n = 50, 150$, respectively). *Table below:* compute time in minutes.

only slightly diminished, but predictions are furnished in half the time on aggregate. Again, I remind the reader that this is a little unfair to LIGP, comparing an R-only implementation to `laGP`'s C library. Another reason this timing comparison is not more impressive is that optimizing the inducing point likelihood to obtain local $\hat{\theta}(\mathbf{x}')$, despite being cubic in m rather than n, tends to take more BFGS iterations than the LAGP analog.

Although LIGP methods do not best LAGP (except NN with $n = 50$) on accuracy, it is important to place these RMSEs in context. Horizontal dashed lines in the left panel of Figure 3.9 offer wider historical perspective. Kaufman et al. (2011)'s reported an RMSE of 1.4 (green line; 99% sparse) with $(N, N') = (4000, 500)$ in 17 minutes via compactly supported kernels.

Gramacy and Apley (2015)'s initial LAGP (ALC) implementation improved that to 0.88 (purple line) in 3 minutes, utilizing eight cores. Subsequent improvements in handling larger (less well-conditioned) matrices, and wider `OpenMP` parallelization bring us to the orders of magnitude more accurate and fast results in Figure 3.9.

More recently a method called SVecchia (Katzfuss et al., 2020), adapted from geostatistics to computer surrogate modeling, has yielded impressive RMSEs of 0.016 (orange line) in similar exercises ($(N, N') = (100000, 20000)$) in about 5 minutes – combining training (4.4 minutes) and testing (0.4 minutes) phases – in a single-core setting. I see this new vanguard of methods as equivalent on the borehole problem, with nuance depending on the application. For example, if you need a one-off prediction, LAGP methods (e.g., ALC) are best, furnishing accurate predictions in fractions of a second without an explicit training phase. With modest testing sizes, LIGP methods are faster when amortizing the cost of template calculation. For larger testing sets, SVecchia methods seem attractive.

Lastly, consider comparing to a more traditional global form of inducing point prediction (Section 3.2.3). Using an LHS for $\mathbf{\Psi}_M$ with $M = 80$ in $[0, 1]^8$ requires only 0.56 minutes to produce predictions (3.6) with fixed lengthscale θ, less than even the space-filling template variations of LIGP. Accuracy is tightly coupled to θ, but MLEs render the method uncompetitive as a single evaluation of the log-likelihood (3.5) takes 9 minutes.

3.5.3 Robot arm

The SARCOS data is a popular computer simulation benchmark from the machine learning literature (Rasmussen and Williams, 2006; Vijayakumar and Schaal, 2000). The data/simulations[8] model seven torque outputs as a function of 21 input variables consisting of position,

[8]Original MATLAB: `http://www.gaussianprocess.org/gpml/data/`; plain text in my Git repo.

velocity, and acceleration of a robot arm. It comes pre-partitioned into a training set of size $N = 44484$ and a testing set of size $N' = 4449$. Here I consider only the first torque output. High input dimensionality and non-uniform design – inputs lie on a low-dimensional manifold in the input space – present surrogates with unique challenges.

One implication of the non-uniform design for LIGP is that a hyperrectangle surrounding $\mathbf{X}_n(\check{\mathbf{x}})$, for median input $\check{\mathbf{x}}$, does not place $\check{\mathbf{x}}$ in its center. Consequently a cHR template would yield an un-centered $\mathbf{\Psi}_m(\mathbf{x}')$. Space-fillingness is preserved, albeit with many points outside of the hypersphere enclosing $\mathbf{X}_n(\check{\mathbf{x}})$. A qNorm template, by contrast, can preserve centering through Φ^{-1}. However, in both cases the low-dimensional input manifold may result in a fair number of inducing points without many $\mathbf{X}_n(\mathbf{x}')$ nearby.

As with previous examples, I perform an input pre-scaling based on separable lengthscales estimated via MLE from a size $n = 1000$ random data subset. After pre-scaling I find that local likelihoods, for both LAGP and LIGP, are flat for many $\mathbf{x}'$, yielding exceedingly long local lengthscales $\hat{\theta}(\mathbf{x}')$ and "washed out" local surrogates. Apparently, in 21 input dimensions, small neighborhoods ($n = 50$ and $n = 200$) provide insufficient information about local lengthscales, i.e., beyond the global one. Although I show results with LAGP in both variations, with and without local MLE calculations (with both isotropic and separable local kernels), all variations entertained perform much better with a fixed $\theta_0 = 1$ for all local calculations.

Figure 3.10 summarizes those results, plotting log RMSE against log computation time. Working from the top of the figure (lowest predictive accuracy) downward, observe that default LAGP (blue), i.e., with local MLE lengthscales, performs worst. Larger local neighborhoods ($n = 200$ vs. $n = 50$) do not help accuracy much, and hurt speed. Separable lengthscales improve accuracy by an order of magnitude, but you do even better by sticking with a fixed $\theta_0 = 1$ after pre-scaling, which brings us to the second (red) group. Foregoing

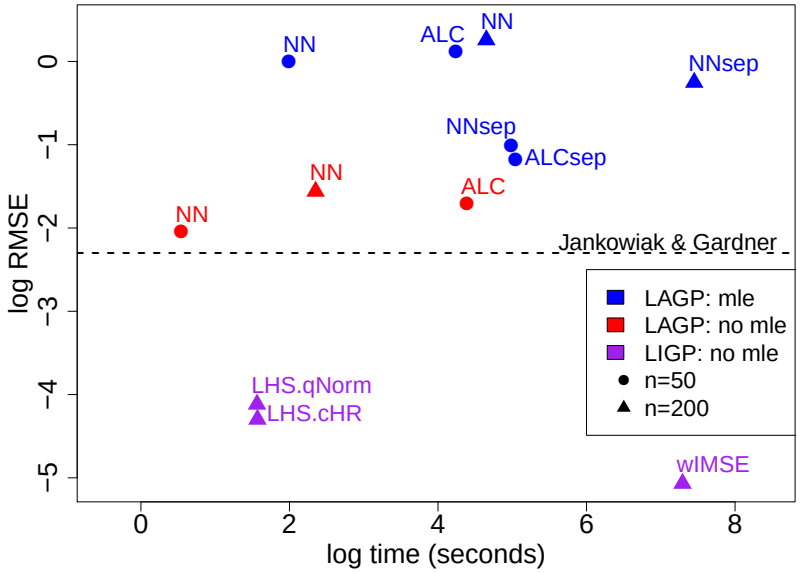


Figure 3.10: LAGP v. LIGP models pitting log RMSE (y-axis) against log time (x-axis) on SARCOS data. LAGP fits included both isotropic and anisotropic (sep) local lengthscales. Fixing local $\theta_0 = 1$ (no mle) yields computational and predictive advantages.

local MLE calculation conveys a several orders-of-magnitude speed-up. These RMSEs are on par with the best methods in recent studies. For example, Jankowiak and Gardner (2019) report on a bakeoff of ten deep and shallow GP and neural network comparators, with best RMSE of 0.107, which in log space is -2.3 (dashed horizontal line).[9] Keeping it simple in high dimension, especially when the training data lie on a lower-dimensional manifold, helps control estimation risk and enhances stability. Larger neighborhoods give a small accuracy fillip, but substantial increase in computation time.

Finally, LIGP methods $(m, n) = (80, 200)$ fall into the last/lowest (purple) group with the highest accuracy. These are 4-5 orders of magnitude more accurate than the default LAGP setup, 2-3 orders better than nomle-LAGP. Compute times are commensurate with the red/middle group, excepting two cases. An wIMSE template pays accuracy dividends for increased computational cost. Simple LAGP (NN) is faster but substantially less accurate.

[9] No timings provided; the worst method had RMSE 0.25.

I again remind that these timings are unfair to LIGP's R-only implementation.

3.5.4 Satellite Drag

Finally, consider large data sets of simulated drag coefficients for satellites in low-Earth orbit. For a description of these data see Sun et al. (2019a), Mehta et al. (2014), Gramacy (2020, Chapter 2.3.3) and the Git repo `https://bitbucket.org/gramacylab/tpm/src`. I seek accurate surrogates for drag for the Hubble Space Telescope (HST). Simulations, via so-called test particle MC (TPMC), treat atmospheric elements of atomic oxygen (O), molecular oxygen (O_2), atomic nitrogen (N), molecular nitrogen (N_2), helium (He), or hydrogen (H) separately. Following previous studies, I consider surrogates for these "species" separately. Data for each species is comprised of a two million-sized (N) LHS over eight configuration inputs. The goal is to predict drag to a 1% relative RMSE (RMSPE) accuracy. Big training data are essential to meeting that benchmark, and needless to say ordinary large-N GP surrogates are not a viable alternative.

Figure 3.11 summarizes the results of tenfold cross-validation for each species. The 1% benchmark is shown horizontally at zero in log space. Again mimicking previous experiments, I pre-scale (Section 3.5.1) after coding inputs and before fitting local approximations. Observe in the left panel that LAGP (NN) with $n = 150$ is the only method able to produce log RMSPEs below the 1% benchmark for all folds. However, LIGP (wIMSE) and LIGP (qNorm) come in at a close second and third and have medians (over all folds) below the 1% benchmark. Factoring in computation time (right panel), LIGP methods predict roughly 50% faster than LAGP (NN) with $n = 150$. Given the scale of the test and training sets, even LIGP (wIMSE) emerges as a viable, cheap alternative.

In contrast to the previous two examples, LIGP (cHR) accuracy suffers relative to the

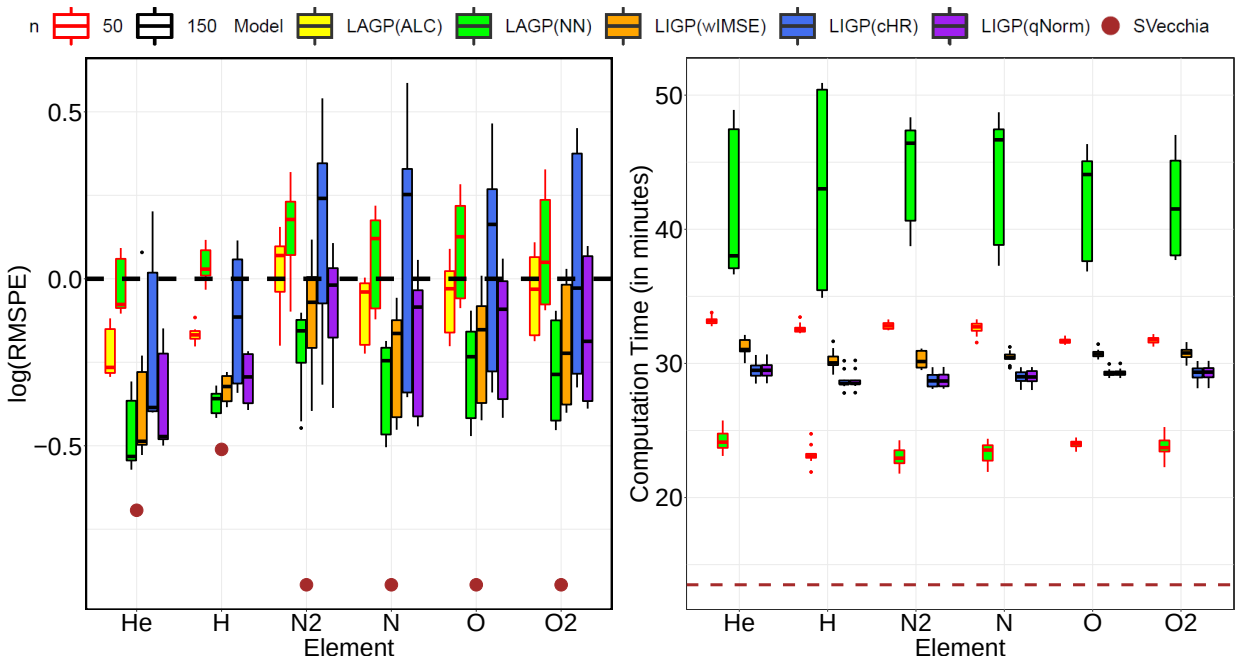


Figure 3.11: *Left:* accuracy over tenfold cross validation for each species via log RMSPE. The horizontal line denotes the 1% benchmark in log space. *Right:* prediction compute time (in minutes) across cross-validation folds.

other space-filling template scheme LIGP (qNorm). This may be due to non-stationarity. Inducing points that lie within the neighborhood – thus motivating LIGP (qNorm) – transfer more of the flexible structure of the GP and provide more accurate predictions. Finally, results recently released using SVecchia (brown) offer further improvement, although only when substantial training time is amortized over a large predictive set. In cases when a single or a relatively small number of predictions are needed, LIGP/LAGP can furnish accurate predictions in seconds, whereas SVecchia requires (tens of) minutes.

3.6 Discussion

Exponential growth of diversity and size of computer simulation campaigns places a heavy burden on GP surrogates. Remaining fast enough to be useful – they cannot be slower

than the simulator they are replacing – but without cutting too many corners in approximation, in order to keep fidelity high to capture non-stationary relationships, requires a nimble approach. Many interesting new methods have come online of late, including inducing points and local approximation. Inducing points address computation time and space head on, but sacrifice on fidelity. Existing likelihood based tools for choosing their multiplicity and location are difficult to wield due to an abundance of local minima. Local approximations (LAGP) perform better in prediction exercises because their criteria more squarely target predictive accuracy. However, they rely on cumbersome discrete search to supplant intractably large conditioning sets.

Here I proposed a hybrid approach: locally induced Gaussian processes (LIGPs). Toward that end, I developed a novel weighted integrated mean-squared error (wIMSE) criterion for selecting inducing points nearby predictive locations of interest. Closed forms for the criteria and derivatives were provided. The key insight here is one of replacing discrete data subset selection (LAGP) with continuous, library-based search via wIMSE through inducing points. My empirical work revealed that such conditioning sets had a highly consistent structure from one predictive location to the next, suggesting that one-off calculations could be reused as a template for other locations of interest.

The result is a new transductive GP learner that is faster than the original, with comparable or improved accuracy in out-of-sample exercises. When LIGP results are less accurate than LAGP, the gaps are narrow and LAGP requires substantially more computation. In some cases, LIGP is orders of magnitude more accurate without demanding more computation. My examples spanned illustrative (2d and 8d with tens and hundreds thousands of points) to high-dimensional benchmarks (21d with non-space-filling design) and real-world simulation (8d and millions of runs).

I see these promising results as providing a solid foundation from which to explore

improvements: from accurate and even faster predictions; to broader application such as in low-signal and even heteroskedastic (Binois et al., 2018b) stochastic simulation experiments. I have some specific ideas. Rather than NN neighborhoods for each predictive location, thrifty ALC alternatives (e.g., `alcray` in `laGP`, Gramacy and Haaland, 2016) may enhance the hybrid. Kernel support could be expanded to include other families, such as Matérn, or to include locally separable lengthscales. In addition, automating the choice of local sizes (m, n) through a Bayesian optimization of out-of-sample RMSE could help make the methodology more plug-n-play.

Chapter 4

LIGP for stochastic simulations

4.1 Introduction

Continued advancements in high-performance computing (HPC) and techniques like particle transport and agent-based modeling yield enormous corpora of stochastic simulation data. For a nice review of state-of-the-art simulation and modeling in such contexts, and challenges going forward, see Baker et al. (2020). Examples include simulators for disease/epidemics (Fadikar et al., 2018; Hu and Ludkovski, 2017; Johnson et al., 2018) tumor spread (Ozik et al., 2019), inventory/supply chain management (Hong and Nelson, 2006; Xie and Chen, 2017), ocean circulation (Herbei and Berliner, 2014), radiation/nuclear safety (Werner et al., 2018), and more. In many cases — in particular those cited above — the simulator can exhibit input-dependent-noise, or so-called heteroskedasticity. When the data is noisy, in addition to the usual nonlinear dynamics in mean structure, a large and carefully designed simulation campaign is essential for isolating signal.

And so is modeling. Gaussian process (GP) regression is the canonical choice of surrogate model for simulation experiments because it provides accurate predictions and uncertainty quantification (UQ) which facilitate many downstream tasks such as calibration, input sensitivity analysis, Bayesian optimization, and more. See Santner et al. (2018) or Gramacy (2020) for a review of GPs and for additional context. However, standard GP inference and prediction scales poorly to large data sets. GP modeling involves a multivari-

ate normal (MVN) distribution whose dimension N matches the size of the training data, i.e., $(\mathbf{X}_N, \mathbf{Y}_N)$ for regression. The quadratic cost of storage and cubic cost of decomposition of covariance matrices, for determinants and inverses involved in likelihoods, say, limits N to the small thousands. For many stochastic simulation experiments in modest input dimension, small campaigns are insufficient for learning. Or conversely, larger N experiments cannot be modeled by GPs.

Numerous strategies abound in diverse literatures (machine learning, spatial/geostatistics, computer experiments) to scale up GP capabilities in N. Most rely on approximation. Some take a divide-and-conquer approach, constructing multiple GPs in segments of a partition of the design space (Gramacy and Lee, 2008; Kim et al., 2005; Park and Apley, 2018). A local approximate GP (LAGP; Gramacy and Apley, 2015) offers a kind of infinite-partition, separately constructing $n \ll N$-sized local neighborhood around each testing location $\mathbf{x}'$ in a transductive (Vapnik, 2013) fashion. Small n means thrifty processing despite $\mathcal{O}(n^3)$ complexity. Handling multiple $\mathbf{x}'$ is parallelizable (Gramacy et al., 2014). Predictions are accurate, but a downside to local/partition modeling is that it furnishes an almost pathologically discontinuous predictive surface.

Other approaches target matrix decomposition directly by imposing sparsity on the covariance (Aune et al., 2014; Gardner et al., 2018b; Pleiss et al., 2018; Solin and Särkkä, 2020; Titsias, 2009; Wilson and Nickisch, 2015), the precision matrix (Datta et al., 2016; Katzfuss and Guinness, 2021), or via reduced rank. One rank reduction strategy deploys a set of $M \ll N$ latent knots, or *inducing points*, through which an $\mathcal{O}(M^2N)$ decomposition is derived (e.g. Banerjee et al., 2008; Hoffman et al., 2013; Snelson and Ghahramani, 2006; Williams and Seeger, 2001). In Chapter 3, I hybridized the inducing points approach with LAGP for an extra speed boost/larger neighborhoods n.

GP modeling technology for computer experiments emphasizes the no-noise (i.e., de-

terministic simulation) or constant (iid) noise case, and this remains true when scaling up via approximation. Partition-based schemes are an important exception, as region-based independent modeling imparts a degree of nonstationarity, potentially in both mean and variance. This is not by design in all cases. There are few mechanisms in place to "turn it off", as computational independence (from which speed derives) is tightly coupled to statistical independence (nonstationarity flexibility). LAGP is an exception, as it allows the user to control compute time directly through the neighborhood size, n. Although the software (`laGP`; Gramacy, 2016) supports local inference of so-called nugget hyperparameters, results with noisy data disappoint because small n means it is easy to be fooled into misinterpreting noise as signal, exacerbating predictive discontinuity.

Expressly accommodating input-dependent noise has been explored in the (global/ordinary) GP literature (e.g., Goldberg et al., 1997; Kersting et al., 2007), although this is in the opposite direction on the computational efficiency frontier: introducing N latent nugget variables which must be learned alongside the usual tunable kernel quantities. Stochastic kriging (Ankenman et al., 2010) offers thriftier computation for GP modeling under heteroskedasticity, as long as each of $\bar{N} \ll N$ unique inputs includes a minimum number of runs/outputs; i.e., as long as the per-run degree of replication is high. This allows latent variables to be replaced with independent moment-based variance estimates. Heteroskedastic GPs (HetGP; Binois et al., 2018a) combine these two ideas, using $\bar{N}$ latent variables, leveraging replication in design but not requiring a minimum degree. So-called "Woodbury identities" (Harville, 1998) facilitate (exact, not approximate) GP inference and prediction in $\mathcal{O}(\bar{N}^3)$ time even when N is huge. Binois et al. (2018b) then went on to show how replication, especially in high noise regions of the input space, can be essential for both statistical and computational efficiency. Yet even when $\bar{N} \ll N$, big $\bar{N}$ may be needed to map out the input space, which can be limiting.

In this chapter I propose that, by porting the Woodbury approach from HetGP over to the LIGP framework, one achieves the best of both worlds. Relatively few local-m inducing points could support a much larger number of local unique inputs $\bar{n}$ representing a much larger neighborhood of n total observations under replication. Casting a wide n-net is crucial for separating signal from noise, but would be prohibitive under the original LAGP regime. Bolting a nugget onto LIGP, as is already available in `laGP`, is the tip of the iceberg. Local inducing points, whose number $m \ll \bar{n} \ll n \ll N$ are far fewer, together with Woodbury representations for $\bar{n}$ representing the full n, make for a powerful cascade of local information. I detail the quantities involved for inference, including the gradient for numerical optimization, and prediction. Illustrations are provided along the way, followed by full benchmarking in both constant noise and heteroskedastic settings. I show that this new LIGP capability is both faster and more accurate than modern alternatives.

The remaining flow of the chapter is as follows. An overview of GP regression and LIGP is provided in Section 4.2. My contribution, focused on estimating local noise and incorporating replication in LIGP via Woodbury is detailed in Section 4.3. Section 4.4 summarizes empirical work on synthetic and benchmark examples from epidemiology and biology. Section 4.5 wraps up with a discussion and directions for future work.

4.2 Review

Here I review standard GP surrogate modeling followed by locally induced GPs.

4.2.1 Gaussian process regression

Consider an unknown function $z : \mathbf{X}_N \subset \mathbb{R}^d \to \mathbb{R}$ for a set of d-dimensional design locations $\mathbf{X}_N = (\mathbf{x}_1, ..., \mathbf{x}_N)^\top$ and corresponding noisy observations $\mathbf{Y}_N = (y_1, ..., y_N)^\top$. A common surrogate for such data is a Gaussian process (GP), which places an MVN prior on z. The prior is uniquely defined by a mean vector and covariance (kernel) function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The observation model is $y(\mathbf{x}_i) = z(\mathbf{x}_i) + \varepsilon_i$ where, under constant noise, $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, v)$. Often the iid noise is hyperparameterized by coupling a nugget g with scale τ^2, as in $v = \tau^2 g$ so that the full model for responses is

$$Y_N \sim \mathcal{N}_N(0, \tau^2(\mathbf{K}_N + g\mathbb{I}_N))$$

where $\mathbb{I}_N$ denotes an $N \times N$ identity matrix. Using a zero mean simplifies the exposition without loss of generality.[1] I prefer zero-centered observations and coded/pre-scaled inputs (Wycoff et al., 2021) based on inverse Euclidean distance between rows of $\mathbf{X}_N$. After pre-scaling, kernels k filling out $\mathbf{K}_N$ are reasonable. E.g., the so-called isotropic Gaussian kernel (2.5). A scalar lengthscale hyperparameter θ governs the rate of radial decay of covariance. Other kernel families such as the Matérn (Gramacy, 2020; Stein, 2012, Section 5.3.3), and so-called separable forms with lengthscales for each input coordinate are also common. My work here is largely agnostic to such choices, as long as $k(\cdot, \cdot)$ is differentiable in the coordinates of $\mathbf{x}$. Many of the limiting characteristics of this overly simplistic choice (e.g., infinite smoothness and isotropy) are relaxed by a local application.

Working with MVNs involving dense covariance matrices, as would be furnished by $k(\cdot, \cdot)$, involves storage capacity that grows quadratically in N. Prediction and likelihood evaluation for hyperparameter inference, such as for (τ^2, θ, g), involve inverses and deter-

[1] Any, even nontrivial, mean structure can be subsumed into a covariance kernel if so desired.

minants requiring decomposition (usually via Cholesky) that is cubic in N. To economize on space, the relevant formulas are not provided here as they are not needed later. I shall provide related expressions momentarily in a more ambitious context. These reduce to ones that may be found in textbooks alongside illustrations of many desirable GP properties such as excellent nonlinear predictive accuracy and UQ.

4.2.2 Local approximate GPs

Cubic computational costs are crippling in the modern context of large-N simulation experiments. As a thrifty workaround, Gramacy and Apley (2015) introduced the local approximate GP (LAGP) which imposes an implicitly sparse structure by fitting a separate, limited-data GP for each predictive location $\mathbf{x}'$ of interest. Each fit/prediction is based on an $n \ll N$-sized subset of data $D_n(\mathbf{x}') = (\mathbf{X}_n(\mathbf{x}'), \mathbf{Y}_n(\mathbf{x}'))$ nearby $\mathbf{x}'$. Multiple criteria have been suggested to build this local neighborhood $D_n(\mathbf{x}')$, with Euclidean nearest neighbor (NN) being the simplest. Each prediction thus requires flops in $\mathcal{O}(n^3)$, which can offer dramatic savings and is embarrassingly parallel for each $\mathbf{x}'$ (Gramacy et al., 2014). LAGP was developed for interpolating deterministic simulations, however the `laGP` software on CRAN (Gramacy, 2016) includes a nugget-estimating capability for local smoothing.

This "local nugget" feature underwhelms when applied globally for many $\mathbf{x} \in \mathcal{X}$ comprising a dense testing set. This can be seen first-hand by trying the examples in the `laGP` documentation. Figure 4.1 provides a demo in a stochastic simulation context, involving a stochastic susceptible–infected–recovered (SIR) disease model (Hu and Ludkovski, 2017), implemented as `sirEval` in `hetGP` (Binois and Gramacy, 2018). I used a training set of $\bar{N} = 10000$ unique inputs in $[0,1]^2$, each paired with a random degree of replication in $\{1, \ldots, 20\}$ so that the full experiment comprised of $N \approx 100000$ runs. The views in the

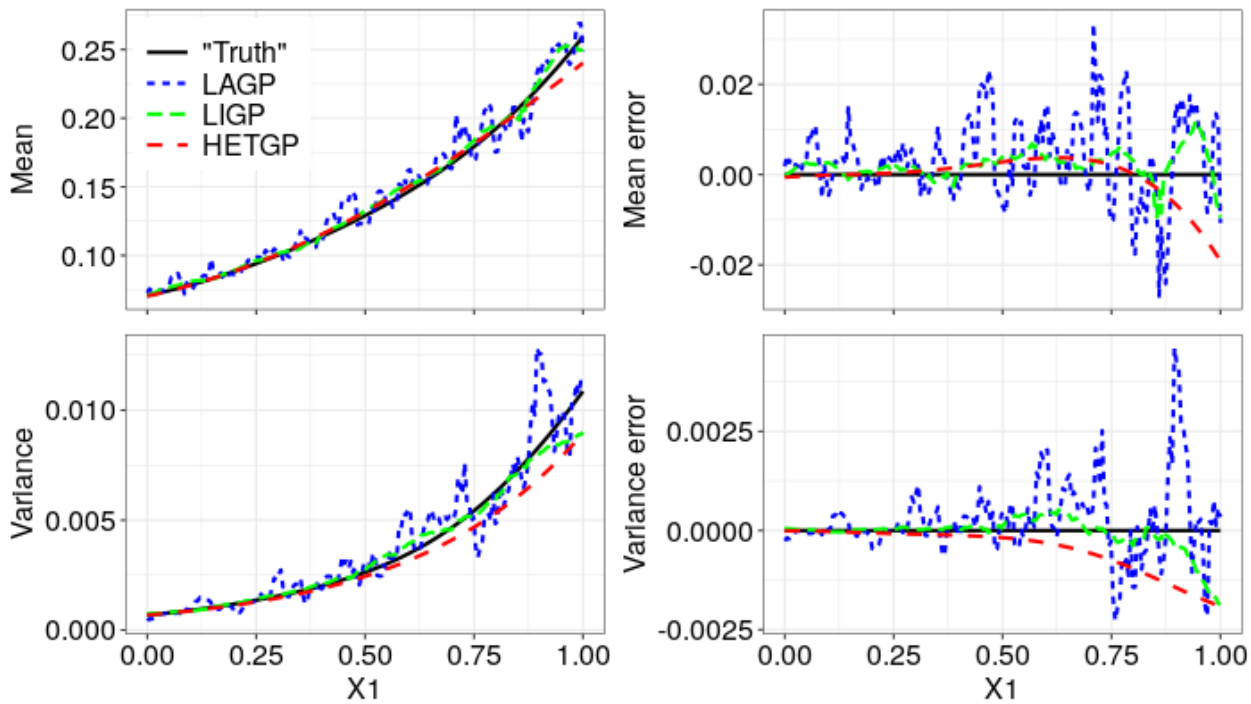


Figure 4.1: Mean (top) and (bottom) variance estimates (left) and errors (right) for SIR model along the slice $x_2 = 0.2$.

figure arise from a grid 200 testing locations along the slice $x_2 = 0.2$. Estimates of the "true" mean and variance (for comparison) are based on smoothed averages of 1000 replicates at each location. Comparators include: LAGP.NN with $n = 50$, LIGP.NN with $\bar{n} = 100$ ($n \approx 1000$) and m=10 inducing points, and HetGP on a random training subset of size $\bar{N} = 1000$. Observe that LAGP's estimates (blue dashed lines) of the mean and variance are highly discontinuous. The error plots in the right column show how LAGP especially struggles to produce accurate estimates for larger x_1. Details on my preferred LIGP method (green dashed line), alongside further analysis of this figure, are the subject of Section 4.3.

The problem with LAGP and noise is both fundamental and technical: fundamental in that you need larger n with noisy data or you'll be fooled by the noise, but this severely cuts into any time advantage; technical in that many stochastic simulation experiments involve replication but the implementation does not adapt its notion of neighborhood accordingly.

In the most extreme suppose, for example, that the nearest input site to $\mathbf{x}'$ has n replicates. Under that configuration LAGP will degenerate without a diversity of unique training inputs. Less pathologically, a multitude of nearby replicates would effectively result in a narrower n-neighborhood even though the number of sufficient statistics is fewer than n. This is inefficient both statistically and computationally. Addressing these two issues is the primary contribution of this chapter.

4.2.3 Locally induced GPs

LIGP in Chapter 3 is an extension of the LAGP idea that relaxes coupling between neighborhood and local covariance structure. Rather than imposing a local covariance structure on $Y_n(\mathbf{x}')$ directly via pairwise distances between $\mathbf{X}_n(\mathbf{x}') \subset \mathbf{X}_N$, LIGP uses an intermediary set of knots, or so-called *inducing points* or *pseudo inputs* $\mathbf{\Psi}_m(\mathbf{x}')$, whose coordinates are not restricted to $\mathbf{X}_N$. This approach is mainstream in the ordinary/global GP approximation context, yielding speedups by reducing the rank of the covariance structure,[2] but its local application is new. When $m \ll n$, the LIGP setup offers a double-speedup, first through neighborhood structure ($n \ll N$) and then through low rank local approximation (cubic in m rather than n). Conversely, it allows a much larger neighborhood n without cubic-in-n increase in flops.

Let $\mathbf{K}_m$ denote a kernel matrix built from inducing points $\mathbf{\Psi}_m(\mathbf{x}')$ and $k_\theta(\cdot,\cdot)$, e.g., in equation (3.1). A discussion of how one may choose $\mathbf{\Psi}_m(\mathbf{x}')$ is deferred to Section 4.3.2. Similarly, write $\mathbf{k}_{nm}$ as cross evaluations of the kernel between $\mathbf{X}_n$ and $\mathbf{\Psi}_m$. Adopting the GP approximation with inducing points introduced by Snelson and Ghahramani (2006) yields

$$\mathbf{\Sigma}_n^{(m)}(\mathbf{x}') = \tau^2 \left(\mathbf{k}_{nm}\mathbf{K}_m^{-1}\mathbf{k}_{nm}^\top + \Delta_n^{(m)} + g\mathbb{I}_n\right) \tag{4.1}$$

[2]Several citations provided in Section 4.1.

as the MVN covariance deployed for inference/prediction at $\mathbf{x}'$. Here $\Delta_n^{(m)} = \text{Diag}\{\mathbf{K}_n - \mathbf{k}_{nm}\mathbf{K}_m^{-1}\mathbf{k}_{nm}^\top\}$ is a diagonal correction matrix for the covariance and $g\mathbb{I}_n$ expresses the pure noise. My notation is suppressing some dependence on $\mathbf{x}'$ to keep the expression tidy. Going forward I shall use $\Omega_n^{(m)} = \Delta_n^{(m)} + g\mathbb{I}_n$ to express the sum of diagonal matrices in equation (4.1), electing not to embolden $\Omega_n^{(m)}$ like I do for other matrices since it can be stored as an n-vector. When $\mathbf{\Psi}_m(\mathbf{x}') \equiv \mathbf{X}_n(\mathbf{x}')$, $\mathbf{\Sigma}_n^{(m)}$ reduces to the standard LAGP covariance $\mathbf{\Sigma}_n(\mathbf{x}') = \tau^2(\mathbf{K}_n + g\mathbb{I}_n)$ and offers no computational benefit. The cost savings comes through the decomposition of $\mathbf{\Sigma}_n^{(m)}$ (4.1) through Woodbury matrix identities. Evaluations of $\mathbf{\Sigma}_n^{-1(m)}$ and $\log|\mathbf{\Sigma}_n^{(m)}|$ may be obtained with $\mathcal{O}(m^2 n)$ flops rather than $\mathcal{O}(n^3)$. Likewise, taking $\mathbf{\Psi}_m = \mathbf{X}_n = \mathbf{X}_N$ with $\mathbf{Y}_n = \mathbf{Y}_N$ yields a global GP without approximation.

Hyperparameter inference may be achieved by maximizing the logarithm of the MVN likelihood $\mathbf{Y}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_n^{(m)})$, which may be expressed up to an additive constant as

$$\begin{aligned}\ell(D_n(\mathbf{x}'), \mathbf{\Psi}_m; \tau^2, \theta, g) \propto -n\log(\tau^2) - \log|\mathbf{Q}_m^{(n)}| + \log|\mathbf{K}_m| - \mathbf{1}_n^\top \log(\Omega_n^{(m)})\mathbf{1}_n \\ - \tau^{-2}\mathbf{Y}_n^\top \left(\Omega_n^{-1(m)} - \Omega_n^{-1(m)}\mathbf{k}_{nm}\mathbf{Q}_m^{-1(n)}\mathbf{k}_{nm}^\top\Omega_n^{-1(m)}\right)\mathbf{Y}_n.\end{aligned} \tag{4.2}$$

Above, $\mathbf{Q}_m^{(n)} = \mathbf{K}_m + \mathbf{k}_{nm}^\top\Omega_n^{-1(m)}\mathbf{k}_{nm}$ and $\mathbf{1}_n$ is a vector of n ones. Differentiating equation (4.2) with respect to τ^2 and solving yields a closed-form MLE $\hat{\tau}^{2(n,m)}$ (deferred until Section 4.3), while a numerical solver like `optim` in R can work with negative concentrated log-likelihood (i.e., plugging $\hat{\tau}^{2(n,m)}$ into equation (4.2)) to estimate $\hat{\theta}^{(n,m)}$ and $\hat{g}^{(n,m)}$. It is important to note that in Chapter 3 I did not include/estimate a local nugget g. This is part of my novel contribution, but I find it expedient to notate quantities here as part of my review.

For fixed hyperparameters $(\hat{\tau}^2, \hat{\theta}, \hat{g})$, a predictive distribution for $Y(\mathbf{x}')$ arises as standard MVN conditioning via an $(n+1)$-dimensional MVN for $(Y(\mathbf{x}'), \mathbf{Y}_n)$. Using $\mathbf{k}_m(\mathbf{x}') =$

$k_\theta(\mathbf{\Psi}_m, \mathbf{x}')$, the moments of that Gaussian distribution are

$$\begin{aligned}\mu_{m,n}(\mathbf{x}') &= \mathbf{k}_m^\top(\mathbf{x}')\mathbf{Q}_m^{-1(n)}\mathbf{k}_{nm}^\top\Omega_n^{-1(m)}\mathbf{Y}_m \\ \sigma_{m,n}^2(\mathbf{x}') &= \hat{\tau}^2\left(k_\theta(\mathbf{x}',\mathbf{x}') + \hat{g} - \mathbf{k}_m^\top(\mathbf{x}')\left(\mathbf{K}_m^{-1} - \mathbf{Q}_m^{-1(n)}\right)\mathbf{k}_m(\mathbf{x}')\right).\end{aligned} \tag{4.3}$$

Both log likelihoods (4.2), and predictive equations (4.3) conditional on those values, reduce to LAGP and ordinary GP counterparts with $\mathbf{\Psi}_m = \mathbf{X}_n$ and $\mathbf{\Psi}_m = \mathbf{X}_n = \mathbf{X}_N$ with $\mathbf{Y}_n = \mathbf{Y}_N$, respectively. Choosing the locations of inducing points $\mathbf{\Psi}_m$, globally or locally, may also be based on the likelihood (Snelson and Ghahramani, 2006), however this is fraught with issues (Titsias, 2009). In Section 3.2.3, I observed that situating $\mathbf{\Psi}_m$ via classical design criteria like integrated variance, or so-called A-optimal design, led to superior out-of-sample predictive performance while also avoiding additional cubic calculation. Specifically for local analysis in LIGP, I introduced a weighted form of $\sigma_{n,m}^2(\mathbf{x}')$, centered around $\mathbf{x}'$, and proposed an active learning scheme minimizing weighted integrated mean-square error (wIMSE) for greedy selection of the next element $\boldsymbol{\psi}_{m+1}$ of $\mathbf{\Psi}_m(\mathbf{x}')$. Upgraded details will be provided in Section 4.3. Saving on computation by stylizing the feature of the local designs for $\mathbf{\Psi}_m(\mathbf{x}')$, Section 3.4.1 developed several template schemes that allowed bespoke optimization of wIMSE for each $\mathbf{x}'$ to be bypassed.

4.3 Local smoothing under replication

LIGP provides an opportunity to reduce computing time while also increasing neighborhood size, because complexity grows cubically in m, not $\bar{n}$ or n. This is doubly important when there are replicates among $\mathbf{X}_N$. Without inducing points, LAGP neighborhoods of size n could have very few unique inputs $\bar{n} \ll n$. However, LIGP's Woodbury representation may

be extended from inducing points to replicates so that neighborhoods of size $\bar{n}$ may be built, regardless of how much bigger n is, without a slowdown.

4.3.1 Fast GP inference under replication

Given a local neighborhood $D_n(\mathbf{x}') = (\mathbf{X}_n(\mathbf{x}'), \mathbf{Y}_n(\mathbf{x}'))$, let $\bar{\mathbf{x}}_i$, $i = 1, \ldots, \bar{n}$ represent the $\bar{n} \ll n$ unique input locations. Notate each unique $\bar{\mathbf{x}}_i$ in $\mathbf{X}_n(\mathbf{x}')$ as having $a_i \geq 1$ replicates, where $\sum_{i=1}^{\bar{n}} a_i = n$. Let $y_i^{(j)}$ be the response of the jth replicate of $\bar{\mathbf{x}}_i$, $j = 1, \ldots, a_i$. Without loss of generality, assume the n-neighborhood inputs are ordered so that $\mathbf{X}_n(\mathbf{x}') \equiv \mathbf{X}_n = (\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_{\bar{n}})^\top$ with each unique input $\bar{\mathbf{x}}_i$ repeated a_i times, and suppose $\mathbf{Y}_n$ is stacked with responses from the a_i replicates in the same order. Based on this construction of D_n, $\mathbf{X}_n = \mathbf{U}\bar{\mathbf{X}}_{\bar{n}}$, where $\mathbf{U}$ is a $n \times \bar{n}$ block matrix $\mathbf{U} = \text{Diag}(\mathbf{1}_{a_1}, \ldots, \mathbf{1}_{a_{\bar{n}}})$. Defining $\mathbf{U}$ in this way, I have $\mathbf{K}_n = \mathbf{U}\mathbf{K}_{\bar{n}}\mathbf{U}^\top$ where $\mathbf{K}_{\bar{n}} = (k_\theta(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j))_{1 \leq, i,j \leq \bar{n}}$, while $\mathbf{U}^\top \Omega_n^{(m)} \mathbf{U} = A_{\bar{n}} \Omega_{\bar{n}}^{(m)}$ where $A_{\bar{n}} = \text{Diag}(a_i, \ldots, a_{\bar{n}})$ and $\Omega_{\bar{n}}^{(m)} = \text{Diag}\{\mathbf{K}_{\bar{n}} - \mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top\} + g\mathbb{I}_{\bar{n}}$.

Using these definitions I present a new expression for $\mathbf{\Sigma}_n^{(m)}$, extending equation (4.1), based on the $\bar{n}$ unique input locations in the local neighborhood:

$$\mathbf{\Sigma}_n^{(m,\bar{n})} = \tau^2(\mathbf{U}\mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top\mathbf{U}^\top + \text{Diag}\{\mathbf{K}_n - \mathbf{U}\mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top\mathbf{U}^\top\} + g\mathbb{I}_n). \tag{4.4}$$

While $\mathbf{\Sigma}_n^{(m,\bar{n})}$ in (4.4) is $n \times n$, I do not build it in practice. It is an implicit, intermediate quantity which I notate here as a step toward efficient likelihood and predictive evaluation.

When $m \ll \bar{n} \ll n \ll N$, decomposing $\mathbf{\Sigma}_n^{(m,\bar{n})}$ to obtain $\mathbf{\Sigma}_n^{-1(m,\bar{n})}$ and $\log|\mathbf{\Sigma}_n^{(m,\bar{n})}|$ is much cheaper than $\mathcal{O}(n^3)$ under the Woodbury identities. Generically, these are as follows

and may be found in most matrix computation resources (e.g., Harville, 1998):

$$(\mathbf{B}+\mathbf{CDE})^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{C}(\mathbf{D}^{-1}+\mathbf{EB}^{-1}\mathbf{C})^{-1}\mathbf{EB}^{-1} \tag{4.5}$$
$$\log|\mathbf{B}+\mathbf{CDE}| = \log|\mathbf{D}^{-1}+\mathbf{EB}^{-1}\mathbf{C}| + \log|\mathbf{D}| + \log|\mathbf{B}|,$$

where $\mathbf{B}$ and $\mathbf{D}$ are invertible matrices of size $n\times n$ and $m\times m$ respectively, and $\mathbf{C}$ and $\mathbf{E}^\top$ are of size $n\times m$. Efficient decomposition in my local approximation context involves combining LIGP's use of equation (4.5) with inducing points, and Binois et al.'s (2018a) separate use leveraging replicate structure. Based on (4.4), let $\mathbf{B} = \mathrm{Diag}\{\mathbf{K}_n - \mathbf{U}\mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top\mathbf{U}^\top\} + g\mathbb{I}_n$, $\mathbf{D} = \mathbf{K}_m^{-1}$ and $\mathbf{E} = \mathbf{k}_{\bar{n}m}^\top\mathbf{U}^\top$. The result is as follows:

$$\mathbf{\Sigma}_n^{-1(m,\bar{n})} = \tau^{-2}\left(\Omega_n^{-1(m)} - \mathbf{U}\Omega_{\bar{n}}^{-1(m)}\mathbf{k}_{\bar{n}m}\mathbf{Q}_m^{-1(\bar{n})}\mathbf{k}_{m\bar{n}}\Omega_{\bar{n}}^{-1(m)}\mathbf{U}^\top\right) \tag{4.6}$$
$$\log|\mathbf{\Sigma}_n^{(m,\bar{n})}| = \log(\tau^2) + \log|\mathbf{Q}_m^{(\bar{n})}| - \log|\mathbf{K}_m| + \sum_{i=1}^{\bar{n}} a_i \log\omega_i^{(\bar{n},m)},$$

where $\mathbf{Q}_m^{(\bar{n})} = \mathbf{K}_m + \mathbf{k}_{\bar{n}m}^\top\Lambda_{\bar{n}}^{(m)}\mathbf{k}_{\bar{n}m}$, $\Lambda_{\bar{n}}^{(m)} = A_{\bar{n}}\Omega_{\bar{n}}^{-1(m)}$, and $\omega_i^{(\bar{n},m)}$ is the ith diagonal term in $\Omega_{\bar{n}}^{(m)}$. Equation (4.6) is key to reducing computational cost from $\mathcal{O}(nm^2)$ in equations (4.2–4.3) to $\mathcal{O}(\bar{n}m^2)$. However again, these quantities are never built directly (especially not $\mathbf{\Sigma}_n^{-1(m,\bar{n})}$).

The representations in (4.6) allow the log-likelihood (4.2) to be expressed as follows

$$\ell(D_n, \mathbf{\Psi}_m; \tau^2, \theta, g) \propto -n\log(\tau^2) - \log|\mathbf{Q}_m^{(\bar{n})}| + \log|\mathbf{K}_m| - \sum_{i=1}^{\bar{n}} a_i \log\omega_i^{(\bar{n},m)} \tag{4.7}$$
$$- \tau^{-2}\left(\mathbf{Y}_n^\top\Omega_n^{-1(m)}\mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^\top\Lambda_{\bar{n}}^{(m)}\mathbf{k}_{\bar{n}m}\mathbf{Q}_m^{-1(\bar{n})}\mathbf{k}_{\bar{n}m}^\top\Lambda_{\bar{n}}^{(m)}\bar{\mathbf{Y}}_{\bar{n}}\right),$$

up to an additive constant. Above, $\bar{\mathbf{Y}}_{\bar{n}}$ stores the averaged responses for each unique row $\mathbf{X}_{\bar{n}}$. Notice that when equation (4.6) is applied to the log-likelihood, the $\mathbf{U}$ terms vanish. In

fact, I do not need this quantity at all – $\mathbf{\Sigma}_n^{-1(m,\bar{n})}$ – except as notational devices. Instead all matrices have dimensions with m and/or $\bar{n}$. Although $\mathbf{Y}_n^\top \Omega_n^{-1(m)} \mathbf{Y}_n$ must still be calculated, this is a product of vectors and the entries in $\Omega_n^{-1(m)}$ may be stored and accessed through $\Omega_{\bar{n}}^{-1(m)}$ and $A_{\bar{n}}$. Their storage and manipulation are thus linear in n.

Differentiating (4.7) with respect to τ^2 and solving, gives the MLE:

$$\hat{\tau}^{2(\bar{n},m)} = n^{-1} \left(\mathbf{Y}_n^\top \Omega_n^{-1(m)} \mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^\top \Lambda_{\bar{n}}^{(m)} \mathbf{k}_{\bar{n}m} \mathbf{Q}_m^{-1(\bar{n})} \mathbf{k}_{\bar{n}m}^\top \Lambda_{\bar{n}}^{(m)} \bar{\mathbf{Y}}_{\bar{n}} \right). \tag{4.8}$$

The equation for $\hat{\tau}^{2(\bar{n},m)}$ in (4.8) still relies on all replicates through the full $\mathbf{Y}_n$ and is equivalent to $\hat{\tau}^{2(n,m)}$ in equation (3.4). This part of the calulation is linear in n. However when replicates are present, its calculation in (4.8) may be much faster owing to a the second term being sized in $\tilde{n}$ and m. Plugging $\hat{\tau}^{2(\bar{n},m)}$ into equation (4.7) yields the following concentrated log likelihood:

$$\begin{aligned} -\ell\left(D_n, \mathbf{\Psi}_m; \theta, g\right) \propto n \log \left(\mathbf{Y}_n^\top \Omega_n^{-1(m)} \mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^\top \Lambda_{\bar{n}}^{(m)} \mathbf{k}_{\bar{n}m} \mathbf{Q}_m^{-1(\bar{n})} \mathbf{k}_{\bar{n}m}^\top \Lambda_{\bar{n}}^{(m)} \bar{\mathbf{Y}}_{\bar{n}} \right) \\ + \log |\mathbf{Q}_m^{(\bar{n})}| - \log |\mathbf{K}_m| + \sum_{i=1}^{\bar{n}} a_i \log \omega_i^{(\bar{n},m)}. \end{aligned} \tag{4.9}$$

The expression above is negated for library-based minimization in search of MLEs $\hat{\theta}^{(\bar{n},m)}$ or $\hat{g}^{(\bar{n},m)}$. Numerical methods such as BFGS (Byrd et al., 1995) work well. Convergence and computing time of such optimizers is aided by closed form derivatives, which are tedious to

derive but are also available in closed form:

$$
\begin{aligned}
-\frac{\partial \ell\left(D_n, \boldsymbol{\Psi}_m, \theta, g\right)}{\partial \cdot} \propto\, & n\left(\mathbf{Y}_n^\top \Omega_n^{-1(m)} \mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^\top \Lambda_{\bar{n}}^{(m)} \mathbf{k}_{\bar{n}m} \mathbf{Q}_m^{-1(\bar{n})} \mathbf{k}_{\bar{n}m}^\top \Lambda_{\bar{n}}^{(m)} \bar{\mathbf{Y}}_{\bar{n}}\right)^{-1} \\
& \times \frac{\partial\left(\mathbf{Y}_n^\top \Omega_n^{-1(m)} \mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^\top \Lambda_{\bar{n}}^{(m)} \mathbf{k}_{\bar{n}m} \mathbf{Q}_m^{-1(\bar{n})} \mathbf{k}_{\bar{n}m}^\top \Lambda_{\bar{n}}^{(m)} \bar{\mathbf{Y}}_{\bar{n}}\right)}{\partial \cdot} \\
& + \operatorname{tr}\left(\mathbf{Q}_m^{-1(\bar{n})} \frac{\partial \mathbf{Q}_m^{(\bar{n})}}{\partial \cdot}\right) - \operatorname{tr}\left(\mathbf{K}_m^{-1} \frac{\partial \mathbf{K}_m}{\partial \cdot}\right) + \sum_{i=1}^{\bar{n}} a_i \frac{\partial \log \omega_i^{(\bar{n},m)}}{\partial \cdot}.
\end{aligned}
\tag{4.10}
$$

Again, observe that none of these log-likelihood-derived quantities (4.7–4.10) involve matrices sized bigger than $m \times \bar{n}$.

Conditioning on the estimated hyperparameters leaves us with the following re-expressions of the LIGP predictive equations (4.3):

$$
\begin{aligned}
\mu_{m,\bar{n}}(\mathbf{x}') &= \mathbf{k}_m^\top(\mathbf{x}') \mathbf{Q}_m^{-1(\bar{n})} \mathbf{k}_{\bar{n}m}^\top \Lambda_{\bar{n}}^{(m)} \bar{\mathbf{Y}}_{\bar{n}} \\
\sigma_{m,\bar{n}}^2(\mathbf{x}') &= \hat{\tau}^{2(\bar{n},m)} \left(k_\theta(\mathbf{x}', \mathbf{x}') + \hat{g}^{(\bar{n},m)} - \mathbf{k}_m^\top(\mathbf{x}') \left(\mathbf{K}_m^{-1} - \mathbf{Q}_m^{-1(\bar{n})}\right) \mathbf{k}_m(\mathbf{x}')\right).
\end{aligned}
\tag{4.11}
$$

Although these look superficially similar to (4.3), the following remarks are noteworthy. Perhaps the biggest difference is that the average of replicates $\bar{\mathbf{Y}}_{\bar{n}}$ is used for the mean. The Woodbury identity ensures that the result is the same as (4.3), both in mean and variance, if the replicate structure is overlooked. In both cases, many of the details are buried in matrices whose $(\bar{n})$ scripts mask a substantial difference in the subroutines that would be required to build the requisite quantities for (4.11) as compared to (4.3).

Kriging with pre-averaged responses $\bar{\mathbf{Y}}_{\bar{n}}$, whether locally or globally, is a common technique for efficiently managing replicates. However, one must take care to (a) adjust the covariance structure appropriately, and (b) utilize all replicates in estimating scale $\hat{\tau}^2$. Binois et al. (2018b) show that $\bar{\mathbf{Y}}_{\bar{n}}$ are not, alone, sufficient statistics, or risk leaving substantial uncertainty un-accounted for in prediction. The Woodbury identities provide that covari-

ance structure (4.4), and the adjustments in n-quantities for an appropriate scale estimate (4.8). Alternative approaches, such as Ankenman et al. (2010)'s stochastic kriging (SK), also use unique $\bar{n}$ inputs and pre-averaged outputs. Asymptotically covering (a), it may be shown that SK yields best linear unbiased predictor (BLUP) when conditioning on estimated hyperparameters. However, the uncertainty in that predictive surface is not fully quantified. For example, SK may only furnish confidence intervals on predictions, not full predictive intervals, because they do not utilize the full set of sufficient statistics (b). Since my Woodbury application links equation (4.11) with (4.3) identically, a BLUP property holds (locally) for LIGP without asymptotic arguments, and with full predictive UQ.

More specifically, consider the following comparison which upgrades an argument from Binois et al. (2018a) to my local inducing points setting. I show how the MLE of the scale parameter τ^2 is not the same (conditioned on θ, g) when calculated original set of data versus the unique design locations with averaged responses. Using unique-$\bar{n}$ calculations based on $\bar{\mathbf{Y}}_{\bar{n}}$ only would result in $\bar{\tau}^{2(\bar{n},m)} = \bar{n}^{-1}\bar{\mathbf{Y}}_{\bar{n}}^\top \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1}\bar{\mathbf{Y}}_{\bar{n}}$ where

$$\bar{\mathbf{\Sigma}}_{\bar{n}}^{(m)} = \mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top + \Delta_{\bar{n}}^{(m)} + gA_{\bar{n}}^{-1} \tag{4.12}$$

weights the nugget (noise) based on the number of replicates a_i at each unique location $\bar{\mathbf{x}}_i$. Whereas my full neighborhood calculation for τ^2 with inducing points in equation (4.8) can be rewritten as follows:

$$\begin{aligned}\hat{\tau}^{2(\bar{n},m)} = N^{-1}\Bigg(&\bar{n}\bar{\tau}^{2(\bar{n},m)} + \mathbf{Y}_n^\top\Omega_n^{-1(m)}\mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^\top\Lambda_{\bar{n}}^{(m)}\bar{\mathbf{Y}}_{\bar{n}} - \\ &\bar{\mathbf{Y}}_{\bar{n}}^\top\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}((A_{\bar{n}}^{-1} - \mathbb{I}_{\bar{n}})^{-1}\Delta_{\bar{n}}^{-1(m)} + \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)})^{-1}\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}\bar{\mathbf{Y}}_{\bar{n}}\Bigg).\end{aligned} \tag{4.13}$$

See the Appendix for details. Observe that $\bar{\tau}^{2(\bar{n},m)}$ above equation (4.12) is but a small part of (4.13). The term following $\hat{\tau}^{2(\bar{n},m)}$ serves as a correction for the variance estimate.

4.3.2 Local neighborhood geography

To dig a little deeper into the differences between LAGP and LIGP, especially as regards handling replicates and LIGP's upgraded capability, consider again the SIR example introduced in Section 4.2.2. Whereas Figure 4.1 involved a continuum of predictions along a 1d slice, Figure 4.2 shows relevant quantities involved in one of the locations $\mathbf{x}' = (0.47, 0.2)$ along that slice, indicated in green. Recall that the input space is in 2d with $\bar{N} = 10000$ unique inputs and replicate degree $a_i \in \{1, 2, \ldots, 20\}$ distributed uniformly. With such a high degree of replication, LIGP's default $n = 50$ neighborhood is small, comprising of only $\bar{n} = 5$ unique locations. As the green dot moves along the slice of Figure 4.1, eventually one of those five will be replaced, resulting in a change in conditioning set of upwards of 1/5 of $\bar{n}$, or 20%. It is perhaps not surprising that the LAGP surfaces in that figure were so "jumpy", and often inaccurate.

LIGP's implementation of Eqs. (4.7–4.11) afford a much larger amount of data in the local model for commensurate computing effort through a template of $m = 10$ inducing points. These are indicated as blue dots in Figure 4.2. The neighborhood of $\bar{n} = 100$ unique design locations are indicated with their number of replicates (black numbers), $n = 1070$ in total. In this instance, LIGP conditions upon 20 times more training data than the LAGP does. The result is more accurate and yields stable mean and variance estimates, shown visually along a slice in Figure 4.1 and summarized in a more expansive Monte Carlo (MC) exercise in Section 4.4.3. Those latter results additionally show that such accurate predictions can be achieved at a computational cost that is substantially lower than LAGP.

The location and spacing of the inducing points $\mathbf{\Psi}_m(\mathbf{x}')$ in Figure 4.2 (blue dots) come from a local space-filling template scheme based on wIMSE. The idea is that a local inducing point set of a desired size can be built up greedily, using integrated variance under a Gaussian

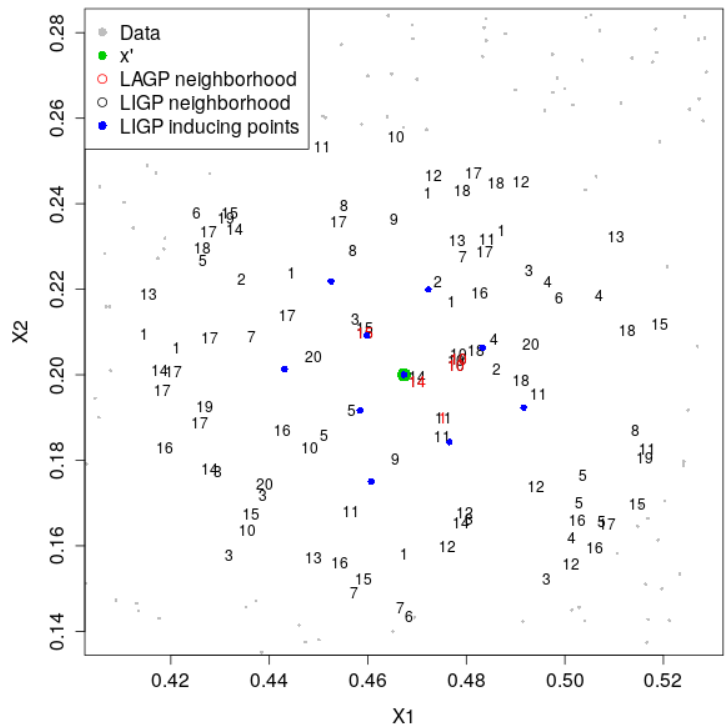


Figure 4.2: LAGP and LIGP local neighborhoods at $\mathbf{x}' = (0.47, 0.2)$, with the numbers denoting the number of replicates at each location. Gray dots represent $\mathbf{X}_N \setminus \mathbf{X}_n$.

measure centered at $\mathbf{x}'$. An expression using $\bar{n}$ unique neighborhood elements (together with all replicates, n) given (4.4) is

$$\begin{aligned} \mathrm{wIMSE}_{\bar{n}}^{(m+1)}(\boldsymbol{\psi}_{m+1}, \mathbf{x}') &= \int_{\tilde{\mathbf{x}} \in \mathcal{X}} k_\theta(\tilde{\mathbf{x}}, \mathbf{x}') \frac{\sigma^2_{m+1,\bar{n}}(\tilde{\mathbf{x}})}{\tau^2} \, d\tilde{\mathbf{x}} \\ &= \frac{\sqrt{\theta\pi}}{2} \prod_{k=1}^{d} \left(\mathrm{erf}\left\{ \frac{\mathbf{x}' - a_k}{\sqrt{\theta}} \right\} - \mathrm{erf}\left\{ \frac{\mathbf{x}' - b_k}{\sqrt{\theta}} \right\} \right) - \mathrm{tr}\left\{ \left(\mathbf{K}_{m+1}^{-1} - \mathbf{Q}_{m+1}^{-1(\bar{n})} \right) \mathbf{W}'_{m+1} \right\}, \end{aligned} \tag{4.14}$$

where erf is the error Gaussian function, a_k, b_k are bounds for a hyperrectangle study region $\mathcal{X} = [a_k, b_k]_{k=1}^d$, and $\mathbf{W}'_{m+1} = \prod_{k=1}^d \mathbf{W}'_{m+1,k}$. Equation (4.14) bears resemblance to equation (3.9), but masks a substantial enhancement in its component parts due to a double-Woodbury application, as notated by $\bar{n}$ scripts. The $(i,j)^{\text{th}}$ entry of $\mathbf{W}'_{m+1,k}$ is $w_{m+1,k}(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)$ defined in equation (3.10). A closed form gradient with respect to $\boldsymbol{\psi}_{m+1}$ may also be de-

rived. As again this is similar in form to equation (3.11), I do not duplicate it here. Given $\mathbf{Q}_{m+1}^{-1(\bar{n})} \equiv \mathbf{Q}_{m+1}^{-1(n)}$, the calculation in (4.14) and derivative are equivalent to equations (3.9)–(3.11). Consequently, a wIMSE optimized design of inducing points is not effected by the local distribution of replicates. Under replication $\mathbf{Q}_{m+1}^{-1(\bar{n})}$ can, of course, be calculated much faster than $\mathbf{Q}_{m+1}^{-1(n)}$, so that optimal configuration can be found quicker. However, it also means that the simple space-filling templates introduced in Section 3.4.2 are equally valid. This is what was used for Figures 4.1–4.2.

Once you have calculated a set of inducing points $\Phi_m(\mathbf{x}')$ for one $\mathbf{x}'$, Section 3.4.1 shows that it can serve as a template for predictions at other members of a testing set $\mathcal{X}$ through displacement and warping so long as the characteristics of the original design $\mathbf{X}_N$ are uniform. Simpler alternatives, based on space-filling criteria also work well. I shall contrast several options in my empirical work in Section 4.4.

4.4 Implementation and benchmarking

Now I provide practical details and report on experimental results showcasing LIGP's potential for superior accuracy and UQ at lower computational costs against LAGP and HetGP. Examples include data from a toy function, as well as real stochastic simulators. All analysis was performed on an eight-core hyperthreaded Intel i9-9900K CPU at 3.60 GHz. Every effort was made to fully tap that distributed resource to minimize compute times.

4.4.1 Implementation details

R code (R Core Team, 2021) supporting all examples reported here and throughout the chapter, in addition to LIGP methodology, may be found on my Git repository.

https://bitbucket.org/gramacylab/ligp/src/master/noise

My main competitors are laGP (Gramacy, 2016) and hetGP (Binois and Gramacy, 2018). While laGP is coded in C with OpenMP for symmetric multiprocessing (SMC) parallelization (R serving only as wrapper), and hetGP leverages substantial RCpp (Eddelbuettel, 2013), my LIGP implementation is pure R using the foreach for SMC distribution. With laGP I use the default neighborhood size of $n = 50$ built by NN and Active Learning Cohn (ALC; Cohn, 1994). For hetGP, I reduce the training data to a random subset of $\bar{N} = 1000$ unique inputs (retaining all replicates) to keep decompositions tractable. Despite compiled C/C++ libraries and thrifty (local/global) data subset choices, my (more accurate and larger-neighborhood) LIGP models are competitive, time-wise, and sometimes notably faster.

My LIGP implementation uses an isotropic Gaussian kernel with scalar lengthscale θ. To improve numerical conditioning of matrices $\mathbf{K}_m$ and $\mathbf{Q}_m^{(n)}$ for stable inversion, I augment their diagonals with $\epsilon_K = 10^{-8}$ and $\epsilon_Q = 10^{-5}$ jitter (Neal, 1998), respectively. My implementation in liGP automatically increases these values for a particular local model if needed for stable decomposition. My LAGP and HetGP results also utilize an isotropic Gaussian kernel. Using separable local formulations do not significantly improve predictive performance, especially after globally prescaling the inputs (Sun et al., 2019a). Pre-scaling or warping of inputs (Wycoff et al., 2021) has become a popular means of boosting predictive performance of sparse GPs (e.g., Katzfuss et al., 2020). In my exercises, I divide by square-root separable global lengthscales obtained from a GP's fit to random $\bar{N} = 1000$ data subsets with averaged responses $\bar{\mathbf{Y}}_{\bar{N}}$ for all fits. See Gramacy (2020), Section 9.3.4, for details. The time required for this, which is negligible relative to other timings, is not included in my summaries.

For inducing point designs $\mathbf{\Psi}_m(\mathbf{x}')$, I use wIMSE templates based on equation (4.14) built at the center of the design space $\check{\mathbf{x}}$, determined as the median of $\mathbf{X}_N$ is each dimension.

For initial local lengthscale $\theta^{(0)}$, I adopt a strategy from `laGP` via 10% quantile of squared pairwise distances between members of $\mathbf{X}_n$.[3] Each $\boldsymbol{\psi}_{m+1}$ augmenting $\boldsymbol{\Psi}_m$ optimizing wIMSE is found via a 20-point multi-start derivative-based L-BFGS-B (Byrd et al., 1995) scheme (using `optim` in R) peppered within the bounding box surrounding the neighborhood $\mathbf{X}_n$ to a tolerance of 0.01. In addition to the wIMSE design for the inducing point template, I consider so-called "qNorm" templates derived from space-filling designs. qNorm templates originate from $m-1$ point Latin hypercube samples (LHS; McKay et al., 2000) through the hyperrectangle enclosing $\mathbf{X}_n(\check{\mathbf{x}})$, augmented with $\check{\mathbf{x}}$ as the mth point. These LHSs are warped with the inverse Gaussian CDF to closely mimic the wIMSE design bypassing the optimization.

Individual local neighborhoods and predictions are made for a set of N' prediction locations $\mathbf{x}' \in \mathcal{X}$ given training data $\{\mathbf{X}_N, \mathbf{Y}_N\}$, neighborhood size $\bar{n}$, and number of inducing points m. I use $\bar{n} = 100$ for each experiment, with $m = 10, 30$ for the 2d and 4d problems respectively. Each location $\mathbf{x}'_i$, for $i = 1, \ldots, N'$ proceeds in parallel via 16 `foreach` threads.[4] To estimate scale and lengthscale I used equations (4.8–4.10) via `optim` through the local neighborhoods of $\mathbf{x}'$. To aid in the delineating signal from noise during hyperparameter optimization, I incorporate default priors for θ and g, from `laGP`, described in Appendix A of Gramacy (2016). Predictions follow equation (4.11).

Finally, each experiment involves same data setup: 30 MC repetitions comprising of novel training sets of $\bar{N} = 10000$ LHS locations, each with $a_i \overset{\text{iid}}{\sim} \text{Unif}\{1, \ldots, 20\}$ for $i = 1, \ldots, \bar{N}$, paired with a separate LHS of size $N' = 1000$ for out-of-sample testing.

[3] In `laGP`, the function providing $\theta^{(0)}$ in this way is `darg`.
[4] Two per hyperthreaded core.

4.4.2 Benchmark non-stationary data

The toy 2d function known as Herbie's tooth (Lee et al., 2011) is attractive as a benchmark problem due to its non-stationary mean surface with multiple local minima. The mean function is defined by $z(x_1, x_2) = -w(x_1)w(x_2)$ where $w(x) = \exp\{-(x-1)^2\} + \exp\{-0.8(x+1)^2\} - 0.05\sin(8(x+0.1))$ and $x_1, x_2 \in [-2, 2]$. For this experiment I introduce constant noise $\epsilon \sim N(0, 0.02^2)$, creating a response $y(x_1, x_2) = z(x_1, x_2) + \epsilon$. My LAGP fits include $n = 100$ via NN for a slightly fairer comparison to LIGP, in addition to the other LAGP options described earlier.

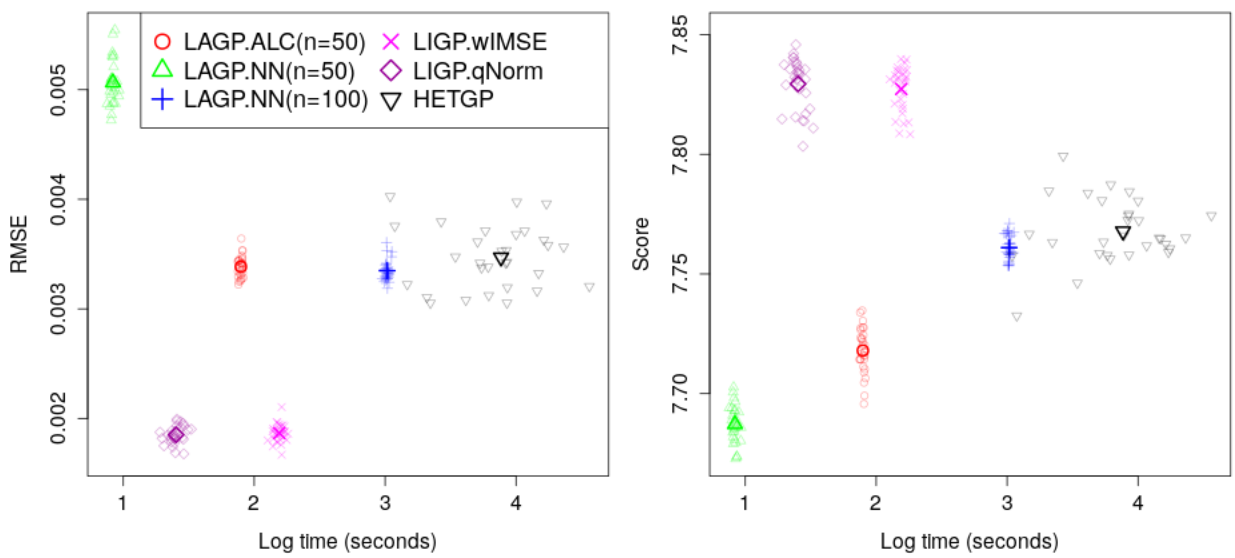


Figure 4.3: RMSE (left) and score (right) vs. log compute time over 30 MC repetitions for Herbie's tooth experiment. Average statistics are denotes with bold markers.

Figure 4.3 shows out-of-sample accuracy statistics versus the logarithm of computation time across the 30 MC repetitions. A particular method's average across the repetitions, taken marginally across both axes, is represented with a bold symbol. The left panel shows the root mean-squared prediction error (RMSE; lower is better) against the (de-noised) truth along the vertical axis. In the right panel, the vertical axis shows proper scores (higher is

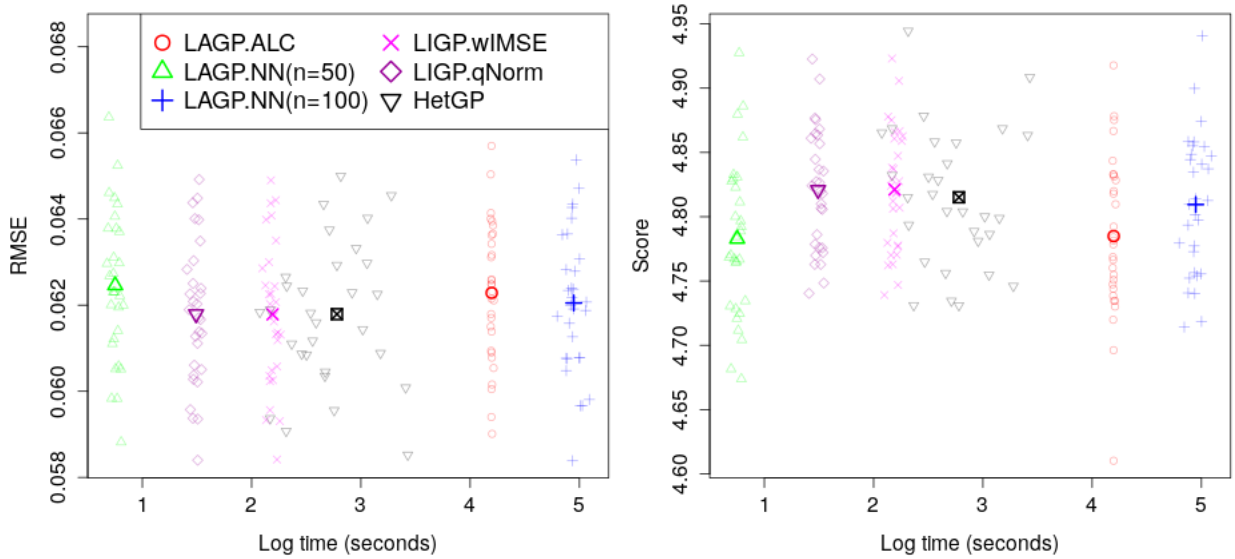
better), combining mean and variance (UQ) accuracy (Gneiting and Raftery, 2007, equation (27)) against the (noisy) test data. LAGP.NN with $n = 50$ (green triangles) is the fastest, but least accurate. LIGP methods yield predictions with the lowest RMSEs and highest scores. LIGP.qNorm (purple diamonds) has the second quickest compute time, a near order of magnitude faster than LIGP.wIMSE due to its thriftier inducing point template design construction. HetGP (black inverted triangle) takes the longest, even via substantial subsetting which explains its high-variance metric on both time and accuracy. It is noteworthy that LAGP.ALC (red circles) and LAGP.NN ($n = 100$) (blue +s) perform similarly for RMSE, but the latter has a higher score. This supports my claim that a wider net of local training data are needed to better estimate noise.

4.4.3 Susceptible-infected-recovered (SIR) epidemic model

I return to the SIR model (Hu and Ludkovski, 2017) first mentioned in Section 4.2.2. SIR models are commonly used for cost-benefit analysis of public health measures to combat the spread of communicable diseases such as influenza or Ebola. I look at a simple model with two inputs: x_1 the initial number susceptible individuals; and x_2 the number of infected individuals. In `hetGP` (Binois and Gramacy, 2018), the function `sirEval` accepts these two inputs on the unit scale. The response of the model is the expected aggregate number of infected-days across Markov chain trajectories until the epidemic ends. The signal-to-noise ratio varies drastically throughout $\mathcal{X}$, making this model an ideal test problem for LIGP.

The out-of-sample accuracy statistics versus the logarithm in computation time displayed in Figure 4.4. The left panel's vertical axis shows the RMSE values between each method's mean predictions and the noisy test data. Variability across repetitions in the data generating mechanism looms large compared to the differences in RMSE values among the

methods. Although RMSE/scores may look similar marginally, within MC repetitions there is a clearer ordering. To expose that, the left section of Figure 4.4's table ranks the methods based on their RMSE (lowest/best first). The reported p-values are for one-sided paired t-tests between the model's RMSEs and those of the next best fit. Although it is a close call, these results indicate that the LIGP (pink ×'s, purple diamonds) and HetGP (black inverted triangles) are significantly lower than those from LAGP.



Model	RMSE	p-value
LIGP.wIMSE	0.061787	0.47
HetGP	0.061790	0.51
LIGP.qNorm	0.061791	<1e-7
LAGP.NN (n=100)	0.06205	<1e-3
LAGP.ALC (n=50)	0.062285	<1e-2
LAGP.NN (n=50)	0.062459	

Model	Score	p-value	Time
LIGP.wIMSE	4.8211	0.41	8.95
LIGP.qNorm	4.8208	0.01	4.45
HetGP	4.8151	0.93	156.13
LAGP.NN (n=100)	4.8095	<1e-5	140.91
LAGP.ALC (n=50)	4.7849	0.66	66.5
LAGP.NN (n=50)	4.7828		2.12

Figure 4.4: *Top:* RMSE (left) and score (right) vs. log compute time over 30 MC repetitions for SIR experiment. Average statistics are denotes with bold markers. *Bottom:* Average RMSEs (left) and scores (right) in ascending rank. The p-values are for one-sided t-tests between the model and the model directly below. Average computation time (in seconds) for each model is listed in the last column.

Score (right panel) offers clearer distinctions between the methods, with the LIGP besting HetGP via the paired t-test below: LIGP.wIMSE and LIGP.qNorm's scores are not

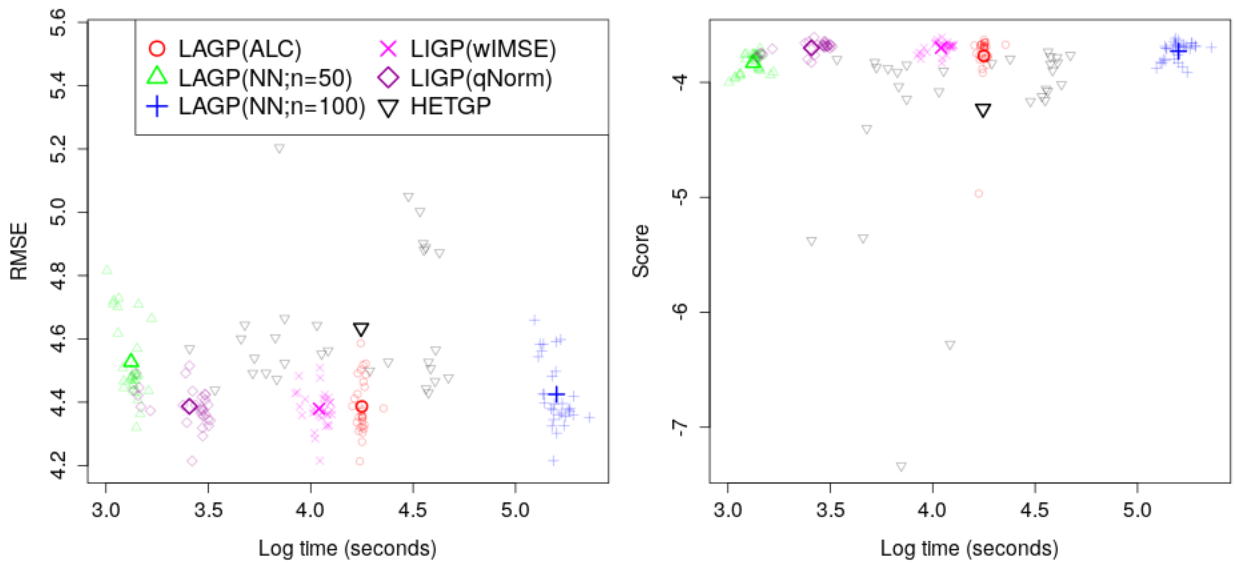
statistically significant from one another (p-value = 0.41), but are significantly higher than HetGP (p-value = 0.01), the next in the list. While the results for HetGP are competitive to LIGP, I remind the reader of the significant time savings LIGP affords, especially LIGP.qNorm, despite being pure R.

4.4.4 Ocean oxygen concentration model

For the final experiment, I apply a stochastic simulator modeling oxygen concentration deep in the ocean (McKeague et al., 2005). This highly heteroskedastic simulator uses MC to approximate the solution of an advection-diffusion equation from computational fluid dynamics. There are four inputs: latitude and longitude coordinates and two diffusion coefficients. The code required to run the simulation can be found in my repository. Inputs are scaled to the unit cube $[0, 1]^4$; the output is the ocean oxygen concentration.

Figure 4.5 shows out-of-sample RMSE and scores for the oxygen concentration experiment. At first glance, HetGP (black inverted triangles) is clearly worse (higher RMSEs, lower scores) than LIGP/LAGP, with considerable more variation across the repetitions. When it comes to modeling this 4d problem, using a subset of only 1000 unique locations does not provide enough information to model this surface. The variation in computation time for HetGP, due to challenges in modeling the latent noise, support this claim.

Among the LIGP and LAGP results, I find LIGP again among the best in RMSE, score, and computation time. LAGP.ALC (red circles) does well at modeling the mean surface (its RMSE average is runner-up to LIGP.wIMSE), but has significantly lower scores that the LIGP methods. LIGP.qNorm (purple diamonds) runs in less than half the time of LAGP.ALC and has comparable RMSE and scores. This continues to support the narrative that LIGP produces equivalent or superior accuracy and UQ among the competitors in a



Model	RMSE	p-value
LIGP.wIMSE	4.3793	0.51
LAGP.ALC (n=50)	4.3865	0.49
LIGP.qNorm	4.3867	<1e-3
LAGP.NN (n=100)	4.4248	<1e-18
LAGP.NN (n=50)	4.5270	0.01
HetGP	4.6346	

Model	Score	p-value	Time
LIGP.wIMSE	-3.698	0.13	56.89
LIGP.qNorm	-3.700	<1e-3	30.19
LAGP.NN (n=100)	-3.729	0.18	181.33
LAGP.ALC (n=50)	-3.769	0.08	70.09
LAGP.NN (n=50)	-3.832	<0.01	22.71
HetGP	-4.23		69.82

Figure 4.5: *Top:* RMSE (left) and score (right) vs. log compute time over 30 MC repetitions for Ocean Oxygen experiment. Average statistics are denotes with bold markers. *Bottom:* Average RMSEs (left) and scores (right) for each model ranked ascending. The p-values are for one-sided t-tests between the model and the model directly below. Average computation time (in seconds) for each model is listed in the last column.

fraction of the time.

4.5 Discussion

Modern computing resources and advances in MC and agent-based modeling strategies have combined of late to produce large simulation campaigns for complex phenomena. The ability to adequately model such large data sets, especially when the simulator is heteroskedastic, is limited. Existing methods for modeling large data sets exist, such as sparse matrix ap-

proximation and divide-and-conquer, are easily fooled and provide inadequate UQ in the presence of input-dependent noise. Local approximations (LAGP) on local subsets of data (neighborhoods) perform well in predictive accuracy for deterministic data, because small local subsets yield a substantial computational advantage over competing methods. Scaling up to larger neighborhoods in the face of common replication-based design strategies for separating signal from noise is inefficient in that context: larger neighborhoods obliterate that computational advantage. LIGP frees up resources through inducing points, allowing increases neighborhood size without severe computational bottlenecks, but is developed solely for deterministic data. HetGP was designed for heterogeneous noisy simulations, and replication in design, and consequently makes thrifty use of sufficient statistics and provides excellent UQ. But it is not a local approximation, and so it does not scale well to the largest simulation campaigns.

Here I provide upgrades to the LIGP framework, essentially by combining these three modern methods, I first redefine LIGP neighborhoods based on the number of unique design locations. Application of the common Woodbury identities allows us to take advantage of replication to perform calculations on the order of the number of inducing points and unique locations. Consequently LIGP may entertain much larger neighborhoods compared to LAGP and earlier versions of LIGP. With these larger neighborhoods, I present results showing LIGP's ability to better separate signal from noise.

The promising results in this chapter provide direction for further areas of study. While current work relies on NN to build local neighborhoods, variance-based alternatives (e.g. `alcray` in `lagp`, Gramacy and Haaland (2016)) may enhance modeling the mean surface. Extending the kernel support to other families, such as Matérn (Stein, 2012) may also prove useful. My empirical work relied on default choices of (m, n) without much exploration. A better understanding of these hyperparameters (possibly through Bayesian optimization of

out-of-sample RMSE or score) could yield better defaults and enhance performance out-of-the-box.

Chapter 5

Entropy-based adaptive design for contour finding

5.1 Introduction

Computer modeling of physical systems must accommodate uncertainty in materials and loading conditions. This input uncertainty translates into a stochastic response from the model, based on nominal settings of a physical system, even when the simulator is deterministic. In engineering, assessing the reliability of said system can mean guarding against a physical collapse, puncture or failing of electronics. Reliability statistics like *failure probability*, the probability the response exceeds a threshold, can be calculated with Monte Carlo (MC). While MC produces an asymptotically unbiased estimator (Robert and Casella, 2013), it can take thousands or even millions of model evaluations, i.e., great computational expense, to achieve a desired error tolerance.

The search for alternatives to direct MC in computer-assisted reliability analysis has become a cottage industry of late. Some approaches seek to gradually reduce the design space for sampling through subset selection (Au and Beck, 2001; Cannamela et al., 2008). *Importance sampling* (IS) focuses MC efforts by biasing sampling toward areas of the design space where failure is probable (Srinivasan, 2013), and then re-weights any expectations to correct for that bias asymptotically. Effective IS strategies (Li et al., 2011; Peherstorfer

et al., 2018a) aim to generate samples which reduce variance compared to direct MC.

A class of *multifidelity* methods leverages a cheaper, low-fidelity computer or surrogate model (Fernández-Godino et al., 2016, 2019; Peherstorfer et al., 2018b). Some variations seek to combine or hybridize information with the expensive high-fidelity analog (Cliffe et al., 2011; Giles, 2008; Litvinenko et al., 2013). Others adaptively trade off between sampling high- and low-fidelty models (Bect et al., 2012; Li and Xiu, 2010; Li et al., 2011). Multifidelity importance sampling (MFIS; Peherstorfer et al., 2016) taps a surrogate to derive a bias distribution for IS, preserving an estimator that is asymptotically unbiased while sparing computational resources. However, a surrogate that is inaccurate around the failure contour can wipe out any potential for computational gains.

Gaussian process (GP) surrogates can help estimate failure probabilities through the experimental design technique known as *active learning* or *adaptive design*, described in Section 5.2.2. In the computer experiments community, many surrogate modelers are concerned with identifying *contours*, also known as level or excursion sets. Most strategies with GPs take a greedy design approach, with new data acquisitions optimizing a selection criteria. Early adoptions modified the traditional Bayesian optimization criterion of expected improvement (Jones et al., 1998) to sample around the contour (Bichon et al., 2008; Picheny et al., 2010; Ranjan et al., 2008). Other criteria such as stepwise uncertainty reduction (SUR) aim to reduce the contour's uncertainty (Bect et al., 2012; Chevalier et al., 2014). Chevalier et al. (2013) and Azzimonti et al. (2016) use random set theory through the Vorob'ev expectation and deviation to minimize the posterior expected distance between the true and estimated contour. Unfortunately, the best of these methods require numerical integration over the input space. Consequently most existing applications involve low-dimensional problems and/or focus on estimating contours that enclose a substantial volume of the design space.

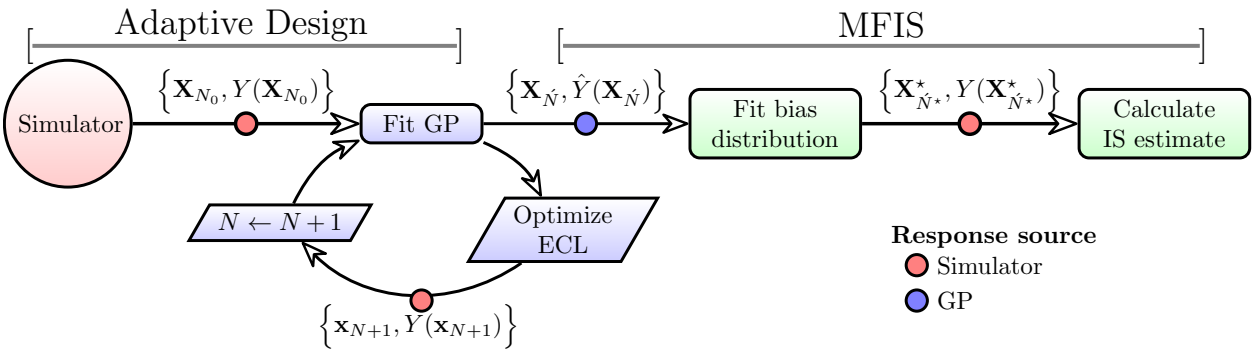


Figure 5.1: Full process combining ECL adaptive design with MFIS. Colored circles represent data with the response generated from the simulator (red) or GP (blue).

I am motivated by reliability applications in engineering that are often exemplified by expensive, high-fidelity simulators and potentially a large number of inputs (large input volume/space). These problems frequently include small failure probability, targeting a contour delineating a failure region occupying less than 0.1% of that space. In addition, I desire a level of conservatism in my analysis. Though many problems contain a single failure region, being conservative includes guarding against missing one or more failure regions. I found that a GP surrogate adaptively designed for contour finding, combined with MFIS, can provide more accurate reliability statistics, with less variability, for fixed sampling effort. Consequently, I introduce a new adaptive design acquisition function based on *entropy* that I call entropy-based contour locator (ECL). Although entropy has been leveraged for active learning with GP surrogates before (e.g. Marques et al., 2018; Oakley, 2004), my setup is distinct. I target more challenging response surfaces with more extreme contours, as motivated by my reliability context. Compared to alternatives, I find that my setup better explores the design space nearby the contours that outline *all* failure regions. My criterion may be calculated in closed form because it does not rely upon numerical integration, allowing it to more easily scale to larger input dimensions. Acknowledging that many expensive simulation campaigns involve parallel evaluation on supercomputers, I extend my criterion to batch selection.

The remainder of the chapter proceeds as follows. In Section 5.2, I provide the necessary background information regarding failure probability estimation, MFIS, and GP adaptive design. I introduce the ECL active learning strategy in Section 5.3. Section 5.4 showcases ECL empirically on a set of benchmark problems. Then I combine ECL designs with MFIS on a more involved and realistic, motivating application in Section 5.5, involving a reliability analysis for a NASA spacesuit design under impact loading. Section 5.6 provides a concluding discussion. Figure 5.1 provides a flowchart for my setup. Although some of the notation in that chart is not yet defined, I include it here because the flow mirrors the chapter development as described above. I encourage the reader to refer back to this figure as related concepts are introduced in Section 5.2. my main contribution (Section 5.3) comprises the cycle in blue; however it is motivated by the green sections, which are showcased on NASA's spacesuit.

5.2 Reliability estimation tools

Here I provide the necessary context for estimating failure probability and contours.

5.2.1 Failure probability estimation

Consider an input–output system, modeled at high-fidelity with $\mathcal{T} : \mathcal{X} \equiv \mathbb{R}^d \to \mathbb{R}$ such that evaluations $\mathcal{T}(\mathbf{x}) \to y$ are computationally/time intensive. Let $\mathbb{F}$ denote a distribution over $\mathbf{x} \in \mathcal{X}$, with density f, representing uncertainties in the inputs $\mathbf{x}$. Reliability is defined through a *limit state function*, $g : \mathbb{R} \to \mathbb{R}$, where failure is deemed to occur when $g(y) > 0$ (Melchers and Beck, 2018). In engineering, a failure is often characterized as an output y exceeding a certain threshold T, expressing the limit state function as $g(y) = y - T$. The

failure region is characterized as

$$\mathcal{G} = \{\mathbf{x} \in \mathcal{X} : g(\mathcal{T}(\mathbf{x})) > 0\}, \quad \text{implying contour} \quad \mathcal{C} = \{\mathbf{x} \in \mathcal{X} : g(\mathcal{T}(\mathbf{x})) = 0\}. \tag{5.1}$$

I use equation (2.25) to represent the probability of failure (α). For technical/theoretical aspects of development in sequential design/active learning literature, the contour of interest $\mathcal{C}$ is sometimes assumed to be the *zero contour*, without loss of generality. However, my interest lies in identifying low/high quantiles of $\mathcal{T}(\mathbf{x})$, corresponding to a small α. My experience is that this setting is more challeging than zero or other more central contours in practice. I believe it is prudent to caution the reader against taking such liberties and arbitrarily describing $\mathcal{C}$. Direct MC sampling is perhaps the most common numerical integration scheme for equation (2.25), yielding approximate estimator $\hat{\alpha}_{\text{MC}}$. With M samples, $\mathbf{x}_i \overset{\text{iid}}{\sim} \mathbb{F}$, $i = 1, \ldots, M$, the MC estimate for α and its associated variance are

$$\hat{\alpha}_{\text{MC}} = \frac{1}{\acute{N}} \sum_{i=1}^{\acute{N}} \mathbb{1}_{\{g(\mathcal{T}(\mathbf{x}_i))>0\}} \qquad \text{with} \qquad \text{Var}\left(\hat{\alpha}_{\text{MC}}\right) = \frac{\hat{\alpha}_{\text{MC}}\left(1 - \hat{\alpha}_{\text{MC}}\right)}{\acute{N}}. \tag{5.2}$$

While $\hat{\alpha}_{\text{MC}}$ in equation (5.2) is asymptotically unbiased (Robert and Casella, 2013), it can take $\acute{N}$ in the millions to achieve a desirable accuracy, which might tax resources required for evaluation of $\mathcal{T}$. For example, if $\alpha = 10^{-4}$ and I wish to have a root mean squared error (RMSE) of 10^{-6} (corresponding to $\text{Var}\left(\hat{\alpha}_{\text{MC}}\right) < 10^{-12}$), $\acute{N} > 10^8$ samples are needed.

One solution to reducing this cost is through a focused simulation like *importance sampling* (IS). With IS, $\mathbf{x}_i^\star \overset{\text{iid}}{\sim} \mathbb{F}_\star$ are generated from an auxiliary distribution biased toward region(s) of inputs more likely to cause failure (i.e., land in $\mathcal{G}$). The resulting estimate

$$\hat{\alpha}_{\text{IS}} = P^{\text{IS}}(\mathbf{x}_1^\star, \ldots, \mathbf{x}_{\acute{N}^\star}^\star) = \frac{1}{\acute{N}^\star} \sum_{i=1}^{\acute{N}^\star} \mathbb{1}_{\{g(\mathcal{T}(\mathbf{x}_i^\star))>0\}} w(\mathbf{x}_i^\star) \quad \text{via weights} \quad w(\mathbf{x}^\star) = \frac{f(\mathbf{x}_i^\star)}{f_\star(\mathbf{x}_i^\star)}, \tag{5.3}$$

is asymptotically unbiased given $\text{supp}(f) \subseteq \text{supp}(f_\star)$ (Robert and Casella, 2013, Section 14.2). Constructing *bias distribution* $\mathbb{F}_\star$ can be expensive in its own right, motivating the search for a cheaper low-fidelity model, like a surrogate $\mathcal{S}$.

Multifidelity importance sampling (MFIS) combines the low cost of generating samples from $\mathcal{S}$ when training the bias distribution in IS (Peherstorfer et al., 2016). Algorithm 5 shows pseudocode for MFIS: a bias distribution is generated by first evaluating $\acute{N}$ samples with the cheap surrogate model $\mathcal{S}_N$, foreshadowing my notation for a GP surrogate (Section 5.2.2) which is trained on N samples $\mathbf{X}_N \in \mathcal{X}$ paired with high-fidelity outputs $\mathbf{Y}_N$. Using the limit state function g, the samples that are classified to produce a failure response are used to train the bias distribution $\mathbb{F}_\star$ for IS. Peherstorfer et al. (2016) suggest using a Gaussian mixture model (GMM; Celeux and Govaert, 1995) for $\mathbb{F}_\star$. A set of $\acute{N}^\star$ samples is drawn from $\mathbb{F}_\star$ and used to evaluate the high-fidelity model. The associated responses $\mathbf{Y}_{\acute{N}^\star}$ are used along with IS weights $\mathbf{w}_{\acute{N}^\star}$ to yield failure probability estimate $\hat{\alpha}^{\text{MFIS}}$.

Algorithm 5 Multifidelity Importance sampling

procedure MFIS($\mathcal{T}$, $\mathcal{S}_N$, $\mathbb{F}$, $\mathbb{F}_\star$, $\acute{N}$, $\acute{N}^\star$, κ)
  Draw $\acute{N}$ samples $\mathbf{X}_{\acute{N}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{\acute{N}}\}$ from $\mathbb{F}$
  $\hat{\mathbf{Y}}_{\acute{N}} \leftarrow \mathcal{S}_N(\mathbf{X}_{\acute{N}})$ ▷ Predictions from surrogate
  Train a GMM of κ clusters with set $\mathcal{F} = \{\mathbf{x} \in \mathcal{X} : g(\hat{y}(\mathbf{x})) > 0\}$ ▷ Form $\mathbb{F}_\star$
  Draw $\acute{N}^\star$ samples $\mathbf{X}^\star_{\acute{N}^\star} = \{\mathbf{x}^\star_1, \ldots, \mathbf{x}^\star_{\acute{N}^\star}\}$ from $\mathbb{F}_\star$
  $\mathbf{w}_{\acute{N}^\star} = \left[\frac{f(\mathbf{x}^\star_1)}{f_\star(\mathbf{x}^\star_1)}, \ldots, \frac{f(\mathbf{x}^\star_{\acute{N}^\star})}{f_\star(\mathbf{x}^\star_{\acute{N}^\star})}\right]$ ▷ Compute importance weights
  $\mathbf{Y}_{\acute{N}^\star} \leftarrow \mathcal{T}(\mathbf{X}^\star_{\acute{N}^\star})$ ▷ Evaluate high-fidelity model
  $\hat{\alpha}_{\text{MFIS}} = \frac{1}{\acute{N}^\star} \sum_{i=1}^{\acute{N}^\star} \mathbb{I}_{g(y_i)>0} w(\mathbf{x}^\star_i)$ ▷ MFIS estimate through equation (5.3)
  return $\hat{\alpha}_{\text{MFIS}}$
end procedure

MFIS seeks to preserve the unbiasedness of the estimator in equation (5.3) while minimizing the number of high-fidelity evaluations $\acute{N}^\star$. Yet the question of how to train a $\mathcal{S}_N$ for MFIS is not widely discussed in the literature. As a simple experiment, I

conduct MFIS with the GP $\mathcal{S}_N$ on the Ishigami function (Ishigami and Homma, 1990): $\mathcal{T}^{\text{Ish}}(\mathbf{x}) = \sin(x_1) + 5\sin^2(x_2) + 0.1x_3^4\sin(x_1)$ for $\mathbf{x} \in [-\pi, \pi]^3$. $\mathcal{T}^{\text{Ish}}$ is smooth, with six global minima creating six disjoint failure regions. I define the failure region $\mathcal{G}^{\text{Ish}}$ for values y below the threshold $T = -9.2244$. To align this with my definition in equation (5.1), let $g^{\text{Ish}}(y) = -y + 9.2244$, which corresponds to $\alpha^{\text{Ish}} \approx 0.001$. In my experiment, I studied estimates of α^{Ish} for increasing budgets of simulator evaluations in an MC exercise with fifty repeats.

The left panel of Figure 5.2 shows errors $|\hat{\alpha} - \alpha^{\text{Ish}}|$ for three comparators with the dark lines indicating means, and 95% confidence interval (CI) as shaded bands. A simple MC estimate $\hat{\alpha}_{\text{MC}}$ from equation (5.2) is along the green dotted line; observe that its absolute error decreases as the number of high-fidelity samples ($\acute{N}$ on the x-axis) increases, showing a slow convergence to the truth (zero error). The blue dashed line represents MFIS-based estimates from equation (5.3) using a GP $\mathcal{S}_N$ trained on $N = 200$ space-filling Latin Hypercube samples (LHS; McKay et al., 2000) generating evaluations of the simulator $\mathcal{T}^{\text{Ish}}$, followed by $\acute{N}^\star$ draws from $\mathbb{F}_\star$ (with $N + \acute{N}^\star$ on the x-axis). Finally, the red solid line represents the GP trained via my proposed entropy-based adaptive design (30 LHS followed by 170 acquisitions; details in Section 5.3). Notice that the red solid line starts better than both (small $\acute{N}^\star$) and, unlike the blue dashed line, offers noticeable further improvement (larger $\acute{N}^\star$).

The right panel of Figure 5.2 shows the proportion of the input space that is correctly classified as being inside (black/left y-axis) and outside (magenta/right) the failure region. This calculation is based on a 10^6-sized LHS in the 3d input space. Observe that the LHS-based GP produces considerably inferior estimates. The adaptive method correctly identifies most of failure points (sensitivity) and nearly all of the non-failure points (specificity). GPs trained on space-filling LHSs often miss some of the failure regions, resulting in a much

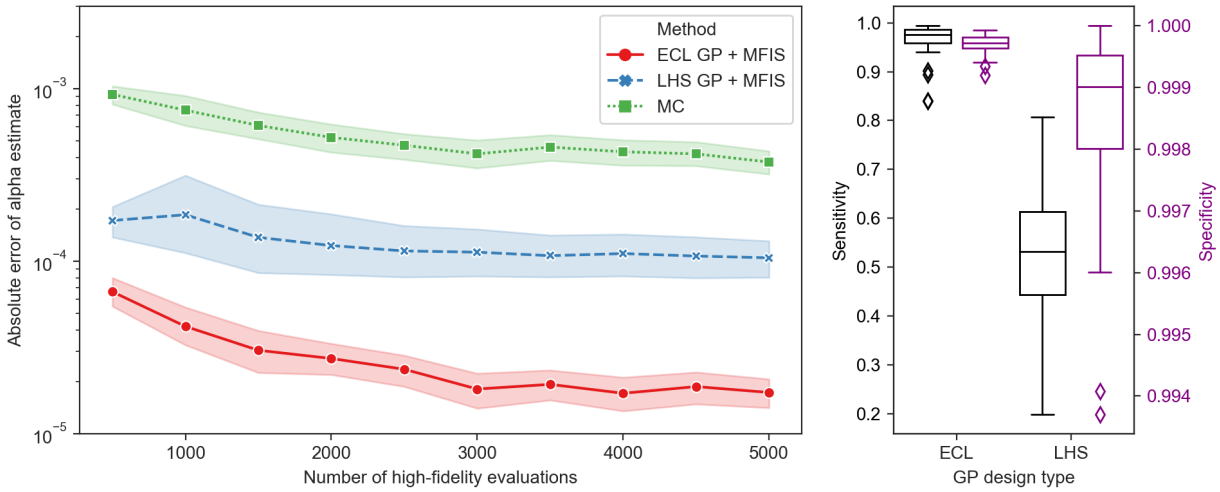


Figure 5.2: *Left:* Absolute errors of failure estimates (truth is 10^{-4}) for the Ishigami function. Bold lines denote mean error over 50 samples with shading for the 95% confidence regions. *Right:* failure region classification statistics across 50 MC repeats for the two GPs.

lower sensitivity. Lower specificity in the LHS option means that more locations that are not failures are being labeled as such. If used for MFIS, the result would be a flawed biasing, leading to inefficient reliability estimates.

5.2.2 Adaptively designed Gaussian processes surrogates

GPs make popular surrogates due to their tremendous flexibility in modeling complex surfaces (Gramacy, 2020; Sacks et al., 1989b; Santner et al., 2018). They may be simply characterized by placing a multivariate normal (MVN) prior on the observations/outputs $\mathbf{Y}_N$. MVNs are uniquely defined by a mean vector μ, often assumed to be zero in the GP regression context for simplicity, and an $N \times N$ covariance matrix $\mathbf{\Sigma}_N$. For N observations included in $D_N = (\mathbf{X}_N, \mathbf{Y}_N)$, the joint model for the responses is $\mathbf{Y}_N \sim \mathcal{N}_N(\mathbf{0}, \mathbf{\Sigma}_N)$. Entries of $\mathbf{\Sigma}_N$ are calculated from a kernel $k_\Xi(\mathbf{x}_i, \mathbf{x}_j)$, which is usually based on inverse distances in the input space. Many expressions of $k_\Xi(\cdot, \cdot)$ exist, including the squared exponential and Matérn families (Stein, 2012). The kernel hyperparameters Ξ, such as τ^2 (scale) and θ

(lengthscale), are estimated by optimizing the MVN log-likelihood.

The true power of a GP lies in its ability to produce accurate predictions by conditioning on observed data. The predictive equations for a new testing location $\mathbf{x}'$ are defined in (2.2) where $\Sigma(\mathbf{x}', \mathbf{X}_N)$ where $\mathbf{k}(\mathbf{x}', \mathbf{X}_N)$ is the vector of cross evaluations of the kernel $k_\Xi(\cdot, \cdot)$ between $\mathbf{x}'$ and $\mathbf{X}_N$. When I write $\mathcal{S}_N$ I am referring to these equations without a particular $\mathbf{x}'$ in mind. A GP's ability to produce fast, accurate predictions has fueled the rise of *active learning* (i.e. sequential design) techniques in the machine learning community (Settles, 2011). The idea is to leverage the current model's information (fit to existing data) to greedily select the next sample/batch of inputs used to gather more data, updating that information. Making such selections relies on the optimization of an *acquisition function*. A litany of acquisition functions serve diverse modeling goals. These include minimizing predictive variance (Cohn, 1994; MacKay, 1992; Seo et al., 2000), finding function optima (Gramacy, 2011; Jones et al., 1998; Wu et al., 2017) via so-called Bayesian optimization (BO), and accurate hyperparameter optimization (Gramacy and Apley, 2015; Zhang et al., 2019).

Targeting level sets

Several acquisition functions target level sets, or *contours*, and are tailored to GP regression. For my purposes, the contour of interest is the boundary of the failure region(s) in equation (5.1). Some acquisition functions focus squarely on sampling where $\mathcal{C}$ is uncertain (Bect et al., 2012; Bichon et al., 2008; Echard et al., 2010; Lee and Jung, 2008; Ranjan et al., 2008). Others are inspired by BO solutions, striking a balance between extrema-seeking and global posterior uncertainty (Bogunovic et al., 2016; Bryan and Schneider, 2008; Gotovos, 2013). When applying contour finding for failure probability estimation, it is vital to identify all disjoint regions within $\mathcal{C}$ (when more than one exist). Therefore, it is highly desirable

for an adaptive design method to promote exploration. Consequently many acquisition functions involve integrating over the domain $\mathcal{X}$ (Bichon et al., 2008; Chevalier et al., 2013, 2014; Picheny et al., 2010; Ranjan et al., 2008). I know of no examples where such integrals that can be evaluated analytically. Numerical schemes, based on reference grids and other quadrature, do not scale well to large dimension d. Some shortcuts are helpful, e.g., Lyu et al. (2018) leverage predictive variance in (2.2) to simplify some aspects. However, challenges remain to produce a method tractable for estimating a small-volume contour in a big $\mathcal{X}$.

A complimentary class of approaches involve identifying $\mathcal{C}$ by mapping to a classification setting (Azzimonti et al., 2016; Bolin and Lindgren, 2015; Gotovos, 2013). A common measure of uncertainty for discrete random variables is *entropy*. Originating from information theory (Cover, 2006), the entropy of a discrete random variable W is measured as in equation (2.30) where $i = 1, \ldots, k$ indexes events w_i, with probability mass $\mathbb{P}(w_i)$. A mapping that would apply in my setting, as a criterion for valuing inputs $\mathbf{x}$, is with $k = 2$ where

$$w_1(\mathbf{x}) \equiv \{\mathbf{x} \in \mathcal{G}_N\} \quad \text{and} \quad w_2(\mathbf{x}) \equiv \{\mathbf{x} \in \mathcal{G}_N^c\}. \tag{5.4}$$

Here, $\mathcal{G}_N$ is an estimate of the failure region in equation (5.1), say via GP $\mathcal{S}_N$ predictive equations (2.2). I shall be more precise momentarily. Abusing notation slightly, this $H(W)$ would highly value inputs $\mathbf{x}$, via larger entropy, which are close to $\mathcal{C}_N$, the (estimated) boundary of $\mathcal{G}_N$, i.e., such that $\mathbb{P}(w_1) \approx \mathbb{P}(w_2) \approx 1/2$, which is sensible.

Oakley (2004) considered entropy for choosing among candidate LHSs in a two-stage design setup. They were targeting output quantiles, like 95%, implicitly defining $\mathcal{C}$ and thus $\mathcal{G}$ and their GP-estimated analogues. I am interested in much larger quantiles, such that any LHS candidate would have low probability of being nearby $\mathcal{G}_N$, mirroring a calculation similar to that of equation (5.2). Contour Location Via Entropy Reduction (CLoVER; Marques et al., 2018) greedily maximizes the reduction in contour entropy through a lookahead

scheme. As in several other methods, CLoVER relies on a set of reference points or knots in $\mathcal{X}$ for numerical interation. But knots are problematic for reasons similar to the LHS candidates above: hard to get enough nearby/inside the failure region. More knots do not help much because the expense of quadrature is squared.

5.3 Entropy-based adaptive design

MFIS holds the potential to leverage a cheaper surrogate to obtain unbiased reliability estimates, but only if the surrogate is accurate around $\mathcal{C}$. Here I propose an entropy-based contour locator (ECL) acquisition function toward that goal.

5.3.1 Entropy acquisition function

After being fit to training data D_N, my GP surrogate $\mathcal{S}_N$ provides a distribution (2.2) for $Y(\mathbf{x})$. In the case of a failure region $\mathcal{G}$ defined by affine limit state function g, the entropy (2.30) with discrete events (5.4) for input $\mathbf{x}$ relative to estimate $\mathcal{G}_N$ via $\mathcal{S}_N$ is

$$\begin{aligned}
&\mathrm{ECL}(\mathbf{x} \mid \mathcal{S}_N, g) = \\
&\quad - \mathbb{P}\Big(g(Y(\mathbf{x})) > 0\Big) \log \mathbb{P}\Big(g(Y(\mathbf{x})) > 0\Big) - \mathbb{P}\Big(g(Y(\mathbf{x})) \leq 0\Big) \log \mathbb{P}\Big(g(Y(\mathbf{x})) \leq 0)\Big) \\
&\quad = - \left(1 - \Phi\left(\frac{\mu_N(\mathbf{x}) - T}{\sigma_N(\mathbf{x})}\right)\right) \log \left(1 - \Phi\left(\frac{\mu_N(\mathbf{x}) - T}{\sigma_N(\mathbf{x})}\right)\right) \\
&\qquad\qquad - \Phi\left(\frac{\mu_N(\mathbf{x}) - T}{\sigma_N(\mathbf{x})}\right) \log \left(\Phi\left(\frac{\mu_N(\mathbf{x}) - T}{\sigma_N(\mathbf{x})}\right)\right),
\end{aligned} \tag{5.5}$$

where Φ is the standard univariate Gaussian CDF. Noting the symmetry in equation (5.5), this would work identically for any $g(y) = a(y - T)$ where $a \in \{-1, 1\}$, meaning that the underlying criteria is unchanged when searching for extremely small, rather than large,

thresholds. Also, ECL trends higher as the GP's predicted mean $\mu_N(\mathbf{x})$ moves closer to the threshold or $\sigma_N(\mathbf{x})$ is higher, facilitating an exploration–exploitation trade off. ECL is maximized when $\mathbb{P}(g(Y(\mathbf{x})) > 0) = \mathbb{P}(g(Y(\mathbf{x})) \leq 0) = 1/2$, targeting $\mathbf{x} \in \mathcal{C}_N \equiv \{\mathbf{x} : g(\mu_N(\mathbf{x})) = 0\}$.

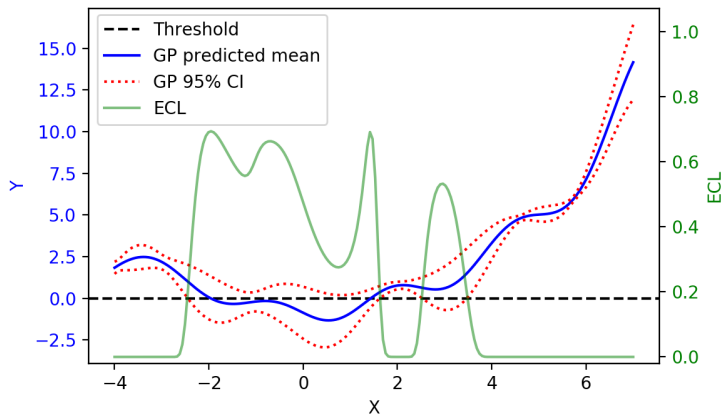


Figure 5.3: GP surrogate predictions and ECL values for the multimodal function along $x_2 = 5.8$.

To explore the ECL surface, consider a 2d multimodal function (Bichon et al., 2008):

$$\mathcal{T}^{\mathrm{MM}}(\mathbf{x}) = \frac{(x_1^2 + 4)(x_2 - 1)}{20} - \sin\left(\frac{5x_1}{2}\right) - 2, \tag{5.6}$$

where $x_1 \in [-4, 7], x_2 \in [-3, 8]$. This function has been used for reliability/level-set finding with $T = 0$ (Bichon et al., 2008; Marques et al., 2018). I trained a GP on an LHS of $N = 20$ points. Figure 5.3 shows the predictive mean ($\mu_N(\mathbf{x})$ blue line), 95% mean confidence interval (roughly $\pm 2\sigma_N(\mathbf{x})$, red dotted lines) and ECL (green line) across the slice $x_2 = 5.8$. Since $g(\mu_N(\mathbf{x}))$ crosses zero twice, there are two global maxima of ECL in this slice alone, along with several other local maxima where the upper bound of the confidence bound is close to or above zero. In higher volume input spaces the dimension of the manifold tracing out the level set also increases, resulting in a continuum of ridges in the ECL surface. If you want to look ahead, Figure 5.4 provides a visual. Such a surface could present challenges to effective

numerical optimization for the purpose of solving for adaptive design acquisitions.

5.3.2 Optimization-based acquisition

To address the (potentially) multimodal nature of ECL in equation (2.30), as illustrated in Figure 5.3, I developed a two-step strategy to solving for each greedy acquisition:

$$\mathbf{x}_{N+1} = \text{argmax}_{\mathbf{x}\in\mathcal{X}} \text{ECL}(\mathbf{x} \mid \mathcal{S}_N).$$

In the first stage I optimize over a discrete-space-filling candidate set $\dot{\mathbf{X}}_{N_c}$. I prefer LHS candidates with $N_c \approx 10d$, following the rule of thumb proposed in Loeppky et al. (2009).[1] Then, in the second stage, I use this global, but coarse, solution to initialize a local solver, such as BFGS (Byrd et al., 1995). The first stage is designed to "select" a global domain of attraction, while the second stage ascends that peak of the ECL surface.

Algorithm 6 Entropy Optimization

1: **procedure** Entropy.Opt$(N_c, \mathcal{S}_N, \mathcal{X})$
2: $\quad\dot{\mathbf{X}}_{N_c} \leftarrow$ LHS of size N_c in $\mathcal{X}$ ▷ Candidate set
3: $\quad\grave{\mathbf{x}} \leftarrow \text{argmax}_{\dot{\mathbf{x}}\in\dot{\mathbf{X}}_{N_c}} \text{ECL}(\dot{\mathbf{x}} \mid \mathcal{S}_N)$ ▷ Discrete search
4: $\quad\mathbf{x}_{N+1} \leftarrow \text{argmax}_{\mathbf{x}\in\mathcal{B}(\grave{x})} \text{ECL}(\mathbf{x} \mid \mathcal{S}_N)$ ▷ Continuous, local, from $\grave{\mathbf{x}}$
5: $\quad$**return** $\mathbf{x}_{N+1}$
6: **end procedure**

The details are laid out in pseudocode in Algorithm 6. In line 4, representing stage 2 local search, the scope is based on $\grave{\mathbf{x}}$. For a true local search, the region $\mathcal{B}(\grave{\mathbf{x}})$ may comprise of a box containing $\grave{\mathbf{x}}$ whose sides extend to nearby elements of the candidate set $\dot{\mathbf{X}}_{N_c}$ or to the edge of $\mathcal{X}$, whichever is closer. Such narrowed scope may help local optimizers that support box constraints, like BFGS does. In my implementation (more details in Section

[1] This chapter targets design sizes for computer experiments, not numbers of active learning candidates, but I find the logic of the two to be quite similar.

5.4.1) simply take $\mathcal{B}(\grave{\mathbf{x}}) = \mathcal{X}$, the whole space, and use $\grave{\mathbf{x}}$ to initialize the search.

I find that the stochastic and coarse global nature of the first stage (owing to the small LHS), paired with a precise and focused second stage, brings benefits beyond avoiding solutions trapped in vastly inferior local maxima. Sometimes the surrogate $\mathcal{S}_N$ becomes over-confident about its predictive equations, leading to small $\hat{\sigma}(\mathbf{x})$ nearby an estimated failure region contour $\mathcal{C}_N$ that is far from the true $\mathcal{C}$. In such situations, the first stage offers potential to ignore that area for local maxima in a less seductive, but almost-as-good region. Injecting a degree of stochasticity into optimization is a convenient way of avoiding pathologies, building-in robustness (Wolpert and Macready, 1997).

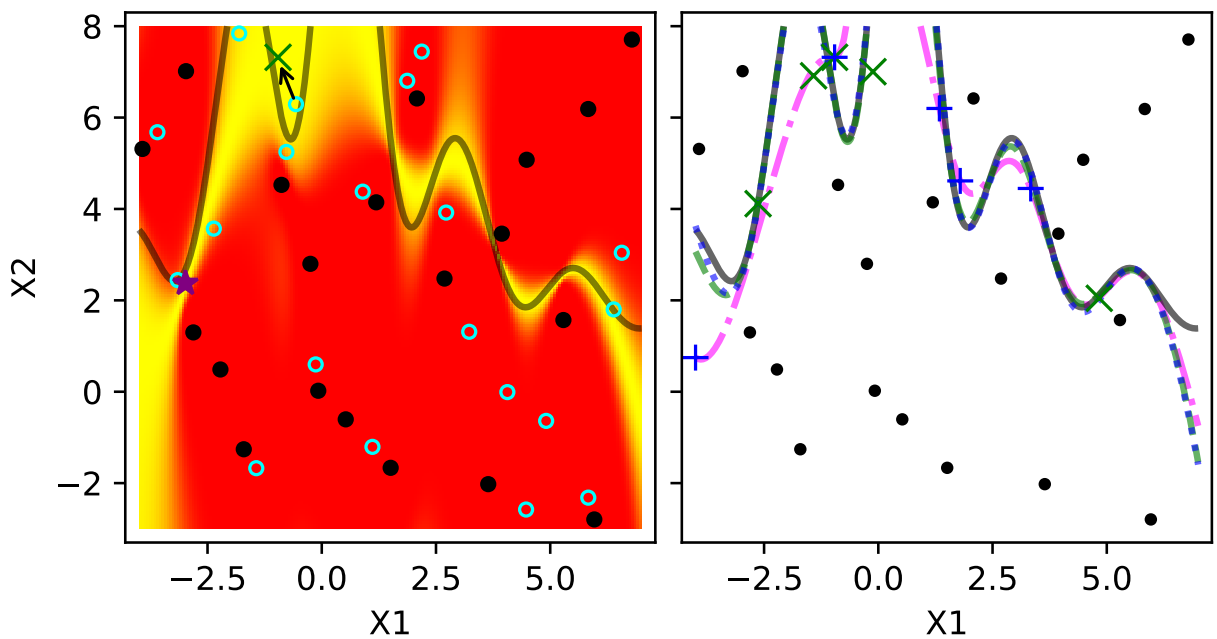


Figure 5.4: *Left:* ECL surface based on a GP fit to $N_0 = 20$ points (black circles). The true $\mathcal{C}^{\text{MM}}$ is denoted by the black curve. Candidates $\mathbf{X}_{N_C}$ are aqua open circles, with $\grave{\mathbf{x}}$ at the terminus of an arrow pointing to $\mathbf{x}_{N_0+1}$, the optimized selection as green '×'. The purple star shows the best of these from all candidates. *Right:* zero contours based on initial GP fit ($\mathcal{S}_{N_0}$ pink dot-dashed curve), at $\mathcal{S}_{N_0+5}$, after five ECL sequentially selected points (green ×'s/dashed), and similarly after a single $N_b = 5$ batch $\mathcal{S}_{N_0+N_b}$ (blue pluses/dotted curve).

An illustration on the 2d multimodal function (5.6) is provided in the left panel of

Figure 5.4. Here I show the ECL surface based on a GP $\mathcal{S}_N$ fit to the same $N = 20$ points as used for Figure 5.3. The multimodal function's true zero contour ($\mathcal{C}^{\text{MM}}$) is shown with the black curve. Observe that the highest ECL regions (yellow) are near that contour. Other local maxima are present where there is little data (bottom left corner) and $\mathcal{S}_N$'s uncertainty high. Aqua blue open circles indicate the candidate set $\dot{\mathbf{X}}_{20}$. The arrow pointing from one blue circle to the green '$\times$' denotes the chosen candidate point $\grave{\mathbf{x}}$ from the first stage and the resulting $\mathbf{x}_{N+1}$ after continuous optimization in the second. Notice that the green '$\times$' is different from the purple star, the optimal location if local optimization is conducted on all 20 candidates. The green '$\times$' finds an almost-as-good region that is close to $\mathcal{C}$.

Although technically (slightly) sub-optimal for the individual acquisition of $\mathbf{x}_{N+1}$, I think that the view in Figure 5.3 (left) suggests that the solution I found (green '$\times$') is indeed sensible. The first stage is more likely to target a high uncertainty region, with a wider domain of attraction (more yellow), due to its coarse nature. Ideally, the attractiveness of "big yellow" regions would be captured formally by the acquisition criteria. This is why most authors integrate over the input space, but that requires numerical quadrature which I am trying to avoid. My two-stage scheme is simpler to implement, and as results shown later in Section 5.4.2 support, fast in execution and robust in a loose sense.

5.3.3 Batch selection

Modern high-performance computing (HPC) resources allow simulations to be run in parallel. To accommodate, here I extend my ECL adaptive design strategy to batch acquisition. Suppose I wish select a batch of N_b samples, e.g., to saturate the node of a computing cluster. One option is to extend my criteria in equation (5.5) from singular $\mathbf{x}$ to multiple $\mathbf{X}_{N_b}$, in the spirit of Zhang et al. (2020a) for variance-based acquisition. This is doable by

upgrading surrogate $\mathcal{S}_N$ pointwise predictive equations in (2.2) to a joint ones. Similar MVN conditioning applies, e.g., see equation (5.3) in Gramacy (2020). Next, calculate entropy in equation (2.30) on the resulting MVN. You will still get a closed form (not shown) up to multivariate CDF evaluations Φ_{N_b}. Calculating Φ_{N_b} can pose a challenge. Although univariate Φ involve a degree of quadrature, evaluation is so fast and accurate (and built-in as native in most programming languages) that I generally regard it as analytic in practice. Multivariate Φ_{N_b}, involving high dimensional quadrature in vast "tail volumes", is much more challenging. See Genz and Bretz (2009). Although software is available for R (Genz et al., 2019), I find that N_b on the scale of the number of cores in supercomputers (e.g., $N_b = 16$ or bigger), evaluation is slow and with accuracy that can be problematic for downstream library-based optimization via methods, e.g., via BFGS.

Instead I prefer to select batch elements sequentially, updating $\mathcal{S}_N \to \mathcal{S}_N^{(+1)}$ to account for each new selected input (but not its response) in the batch until I reach $\mathcal{S}_N^{(+N_b)}$. Specifically, given $0 \leq n_b \leq N_b$ selections so far, I may deduce the following equations for $\mathcal{S}_N^{(+n_b)}$, where $\mathbf{X}_{N+n_b} = \mathbf{X}_N \cup \mathbf{X}_{n_b}$ and with $\mathbf{X}_0 \equiv \emptyset$ reducing to the following using equations in (2.2):

$$\begin{aligned} \mu_N^{(+n_b)}(\mathbf{x}'|\mathbf{X}_{N+n_b}, \mathbf{Y}_N) &= \mu_N(\mathbf{x}'|\mathbf{X}_N, \mathbf{Y}_N) = \Sigma(\mathbf{x}', \mathbf{X}_N)\Sigma(\mathbf{X}_N, \mathbf{X}_N)^{-1}\mathbf{Y}_N \\ \sigma_N^{2(+n_b)}(\mathbf{x}'|\mathbf{X}_{N+n_b}, \mathbf{Y}_N) &= k(\mathbf{x}', \mathbf{x}') - \mathbf{k}(\mathbf{x}', \mathbf{X}_{N+n_b})\mathbf{k}(\mathbf{X}_{N+n_b}, \mathbf{X}_{N+n_b})^{-1}\mathbf{k}(\mathbf{x}', \mathbf{X}_{N+n_b})^\top. \end{aligned} \tag{5.7}$$

Each of these updates $\mathcal{S}_N^{(+n_b)} \to \mathcal{S}_N^{(+n_b+1)}(\mathbf{x}_{n_b+1})$, in particular for $\sigma^{2(+n_b)}(\mathbf{x})$ as $n_b \to n_b + 1$ with new acquisition $\mathbf{x}_{n_b+1}$ augmenting $\mathbf{X}_{n_b}$ to build $\mathbf{X}_{n_b+1}$, can be performed in time quadratic in $N + n_b$ via partition inverse equations (Barnett, 1979; Gramacy, 2020; Gramacy and Polson, 2011, Section 6.3). However, the otherwise cubic burden is often manageable in active learning where the whole enterprise is focused on making N as small as possible.

Algorithm 7 Batch Entropy

1: **procedure** Entropy.Batch($\mathcal{S}_N$, N_b, N_c, $\mathcal{X}$)
2: $\quad\mathcal{S}_N^{(+0)} \leftarrow \mathcal{S}_N$ ▷ Initialize deduced surrogate
3: $\quad$**for** $n_b = 1, \ldots, N_b$ **do**
4: $\quad\quad\mathbf{x}_{n_b} \leftarrow$ Entropy.Opt($\mathcal{S}_N^{(+n_b-1)}, N_c, \mathcal{X}$) ▷ Select next point (Alg. 6)
5: $\quad\quad\mathcal{S}_N^{(+n_b)} \leftarrow \mathcal{S}_N^{(+n_b-1)}(\mathbf{x}_{n_b})$ ▷ Update variance equation (5.7) in GP
6: $\quad$**end for**
7: $\quad$**return** $\mathbf{X}_{N_b} \equiv [\mathbf{x}_1^\top, \ldots, \mathbf{x}_{N_b}^\top]^\top$ ▷ Return batch of N_b design points
8: **end procedure**

With $\mathcal{S}_N^{(+N_b)}$ defined in this way, batch selection becomes an exercise in repeated greedy selection. This is detailed in Algorithm 7, with Algorithm 6 deployed as a subroutine. It is worth remarking that a new candidate set $\dot{\mathbf{X}}_{N_c}$ is generated for each new call of line 4. If I were to rely upon the same candidate set for the entire batch, I would in effect be taking the N_b best candidate points, some of which could be far from the local maxima regions of interest. It is possible (although in practice uncommon) that some of the acquisitions are very close to others in the same batch. Since I operate under the assumption that the high-fidelity model is deterministic, replicates are not helpful. In my implementation (more in Section 5.4.1), I account for this by reverting to the optimal candidate point $\grave{\mathbf{x}}$ for $\mathbf{x}_{n_b}$ in such cases, undoing the stage 2 local search. At the end of the batch, new simulations must be performed at $\mathbf{X}_{N_b}$, forming $\mathbf{Y}_{N_b}$, combining with D_N to obtain D_{N+N_b}. Inference follows for any hyperparameters Ξ required to form $\mathcal{S}_{N+N_b}$ which is, of course, different than the intermediate $S_N^{(+n_b)}$ which may be discarded after each iteration of the loop.

To illustrate, I return to the multimodal example of Section 5.3.2. The right panel of Figure 5.4 compares the acquisition of five runs one-at-a-time (i.e., multiple single acquisitions with updates) and five as a batch. The dash-dotted pink line shows the GP's predicted mean contour via $\mathcal{S}_{N_0}$ before either adaptive design. The green $\times$'s were selected one at a time, leading to the green dashed $\mathcal{C}_{N+5}$ contour. In a similar fashion, the blue $+$'s and dotted curve represent the results from the batch size of 5. Both adaptive designs select

the same first point (close to (-1.25, 7.2)); this is guaranteed by using the same candidate set. In the batch design, only the GP's variance equation is updated, which results in runs being selected close to the original GP's zero contour were ECL is maximized. In contrast, repeated one-at-a-time acquisition allows updating of the mean surface. Yet surprisingly, the resulting contours from both designs (green dashed and blue dotted lines) are fairly similar and capture $\mathcal{C}^{\mathrm{MM}}$ well. This mirrors other work on batch active learning with GPs (Zhang et al., 2020a); little is lost on the batch setting versus its purely greedy analog.

5.4 Implementation and benchmarking

Now I provide results for a series of experiments showcasing ECL, and contrasting against existing contour-finding acquisition functions. I first detail my implementation and the synthetic simulators used in the experiments. Then I present and discuss results of MC sequential design exercises in terms of sensitivity and relative error of the failure region's volume based on GP predictions in (2.2) over acquisition iterations.

All analysis was performed on a six-core hyperthreaded Intel i7-9850H CPU at 2.59GHz. Python and R code (R Core Team, 2021) supporting my methodological contribution, and all examples reported here and throughout the chapter, may be found on my Git repository.

https://bitbucket.org/gramacylab/nasa/src/master/entropy

5.4.1 Implementation and synthetic data

The following implementation details are noteworthy. My methodological contribution is encapsulated in the Python package `eclGP`, incorporating GP functionality from the `sklearn` package (Pedregosa et al., 2011). For all of my synthetic examples, I privilege a separable

Gaussian kernel formulation, although other forms such as Matérn (Stein, 2012) could easily be used. To ensure stability in the calculation of the inverse of the covariance matrix, I fix a jitter (Neal, 1998) of 10^{-6}.

I compare the ECL-based estimates of a contour $\mathcal{C}$ to established contour finding acquisition functions with publicly available code. For CLoVER, I utilize the Python code accompanying Marques et al. (2018), with the only adjustment being setting the candidate and integration knots to LHSs of size 1000 for each experiment (see Section 5.4.3 for a discussion). Implementations for all other methods leverage the R package `KrigInv` (Chevalier et al., 2014), using the `"genoud"` option for optimization and the recommended defaults for constructing integration knots where relevant (SUR, tIMSE). In R, I rely on `DiceKriging` (Roustant et al., 2012) to fit GPs.

Function	$\mathcal{X}$	N_0	N	$g(y)$	vol($\mathcal{G}$)	Quantile
Branin-Hoo	$[-5, 10] \times [0, 15]$	10	30	$y - 206$	2.1783	0.9903
Ishigami	$[-\pi, \pi]^3$	30	200	$-y + 10.244$	0.0250	0.9999
Hartmann-6	$[0, 1]^6$	60	500	$y - 2.63$	0.0011	0.9989

Table 5.1: The design space, allocation of samples, limit state functions, volume of the failure region, and quantiles for the synthetic tests.

Experiments are conducted on the Branin-Hoo (Forrester et al., 2008), Ishigami (Ishigami and Homma, 1990), and Hartmann-6 (Surjanovic and Bingham, 2014)[2] functions. Table 5.1 provides a summary of the experiments, including the number of sample points N and limit state functions $g(\cdot)$ used to define failure regions $\mathcal{G}$. The limit state functions are defined so that $\mathcal{G}$ for each function contains at least two local extremes located in disjoint failure regions. The volume of $\mathcal{G}$ is less than 1% of the total volume of the design region in each instance. Volumes reported in the table are based on averaging 100 MC estimates from LHSs of size 10^6. Dense testing LHSs were used to calculate sensitivity and predicted

[2]following https://www.sfu.ca/~ssurjano/hart6.html

volume of the failure region out-of-sample. For Branin-Hoo an LHS of 5×10^6 points is used for prediction/classification, whereas Ishigami and Hartmann-6 used 10^7. To initialize the sequential design I follow the rule of thumb for setting $N_0, N_c = 10d$ (Loeppky et al., 2009), except for the 2d Branin-Hoo where $N_0 = 10$ was sufficient.

5.4.2 Synthetic tests for contour finding

Earlier in Figure 5.2, when motivating adaptive design strategies for contour location, I showcased results on the Ishigami function. Those showed that using an ECL design of 200 points provides a GP predictive mean that produces much higher sensitivity and specificity than a GP trained on an LHS of the same size. Here I expand that analysis against benchmarks. Figure 5.5 augments with other test functions and the following competitors: EGRA (Bichon et al., 2008), Ranjan (Ranjan et al., 2008), and tMSE (Picheny et al., 2010) as strategies that use closed-form calculations for continuous optimization acquisition; CLoVER (Marques et al., 2018), SUR (Chevalier et al., 2014) and tIMSE (Picheny et al., 2010) require integral approximation.

The figure tracks sensitivity as design size increases (left) and provides a boxplot of the final distribution of sensitivity (middle left). Similarly, relative error for $\text{vol}(\hat{\mathcal{G}})$ is tracked (middle right), followed by its final distribution error (right). Thirty MC repetitions were conducted for each experiment, with the line plots displaying means. Across all three experiments (rows in the figure) my ECL method is among the best achievers in terms of sensitivity (higher is better) and volume error (lower is better).

For Branin-Hoo, ECL and CLoVER – the entropy-based methods – distinguish themselves early on in the adaptive design in both sensitivity and volume error compared to all other methods. By the end of the experiment, all methods have a similar sensitivity. ECL's

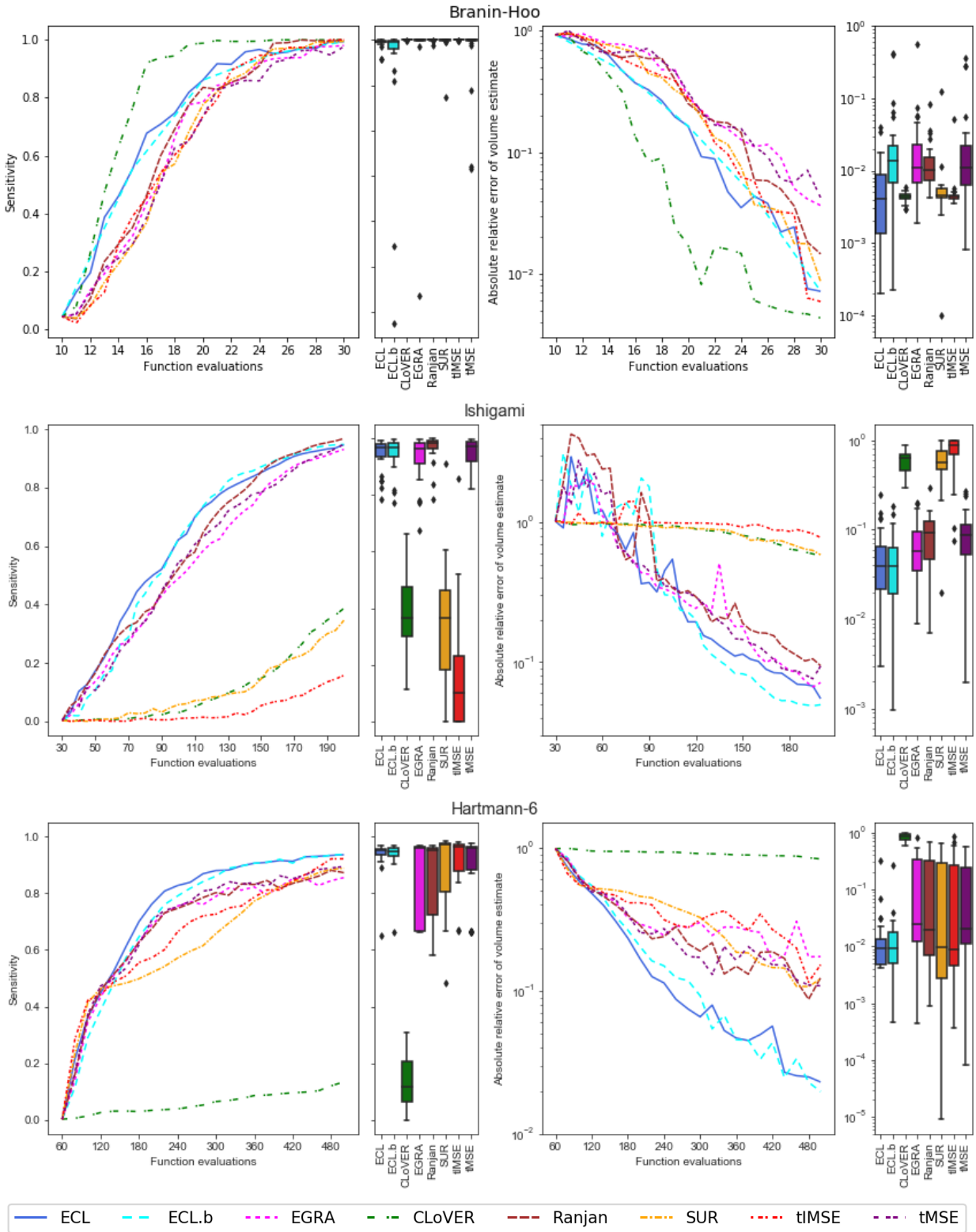


Figure 5.5: Comparing sequential designs in 30 MC repetitions. *Left:* mean sensitivity classifying failure region $\mathcal{G}$. *Middle left:* distributions of sensitivity after last acquisition. *Middle right:* mean relative error $\mathrm{vol}(\mathcal{G})$. *Right:* final distributions of relative volume error.

volume error is in line with the integration methods, beating its closed-form competitors. For Ishigami, observe that final mean sensitivity for ECL is bested by Ranjan. Yet when it comes to the relative error of the contour's volume, ECL is far below that of Ranjan. Hartmann-6 imposes challenges in higher input dimension, which is why I extend the adaptive design to 500 total points. In this experiment, I find that the explorative nature of ECL is key to finding the failure contour. The bottom row of Figure 5.5 shows that ECL provides a much more robust design with high sensitivity and a volume error nearly an order of magnitude smaller than others. I also created a batch ECL design (ECL.b) with batch sizes of 5 for Branin-Hoo and 10 for Ishigami and Hartmann-6. In all three experiments, ECL.b performs similarly to ECL.

For the methods that rely upon a "grid" of integration knots (CLoVER, SUR, tIMSE), the size of that grid greatly affects their ability to explore and identify parts of the failure contour. With a set of 1000 knots for Branin-Hoo, CLoVER performs very well – reaching a high sensitivity plateau before the other methods. Yet when the same number of knots is used in the 3d or 6d problems with a higher quantile, it struggles. In a similar way, the combination of multiple failure regions (six) and a high quantile for the Ishigami experiment poses a great challenge for SUR and tIMSE. See Section 5.4.3 for further discussion.

Method	ECL	ECL.b	CLoVER	EGRA	Ranjan	SUR	tIMSE	tMSE
Branin-Hoo	0.008	0.002	0.42	0.2	0.2	0.4	0.3	0.2
Ishigami	0.10	0.08	17.8	3.9	4.5	3.5	3.4	3.9
Hartmann-6	4.40	0.78	428.4	66.0	59.2	109.1	111.7	59.5

Table 5.2: Average computation times (minutes) to select points across 30 MC repetitions.

Table 5.1 summarizes average computation time to build one full sequential design for each method. The timings include acquisition efforts and subsequent GP updating. Scripts for ECL, ECL.b and CLoVER are in Python and the rest are in R. Note the speed at which ECL and ECL.b adaptive designs are built, especially compared to methods using numerical

quadrature (CLoVER, SUR, and tIMSE). ECL provides between one and two orders of magnitude speedup. Using a batch size of 10 cuts the average computation time of ECL by five for the Hartmann-6 experiment, while still providing designs with similar sensitivity and volume error.

5.4.3 Value of integration knot density

When applying any of the adaptive designs in Section 5.4.2, there are certain choices, or "knobs", that play a crucial role in optimizing the acquisition function. Though both CLoVER and ECL use entropy in their acquisition functions, performing adaptive design with CLoVER includes two knobs, whereas ECL only has one. Here I explore the effect of different knob settings for experiments detailed in Section 5.4.

To begin, I focus on the strategy used to optimize the acquisition function. ECL uses Algorithm 6, which contains a small set of $N_c = 10d$ candidate points and a continuous optimization initialized at the best candidate. For CLoVER, Marques et al. (2018) propose candidates only, forgoing a continuous optimizer. In their code, they use a single candidate set from which to select all new design locations. For my experiments, I fix that set to be an LHS of 1000 points. The results shown in Section 5.4.2 use these strategies. In Figure 5.6, I supplement that experiment with swapped optimization strategies for ECL and CLoVER. ECL's original strategy is denoted with solid lines and triangles and CLoVER's single candidate set strategy is shown with the dashed lines and squares. CLoVER and ECL follow a similar trajectory for sensitivity and relative error when a single candidate set is used. With Algorithm 6, CLoVER's sensitivity lags behind ECL during the first half of the experiment.

In the Ishigami experiment, I find a more complex story (Figure 5.7). Here I provide

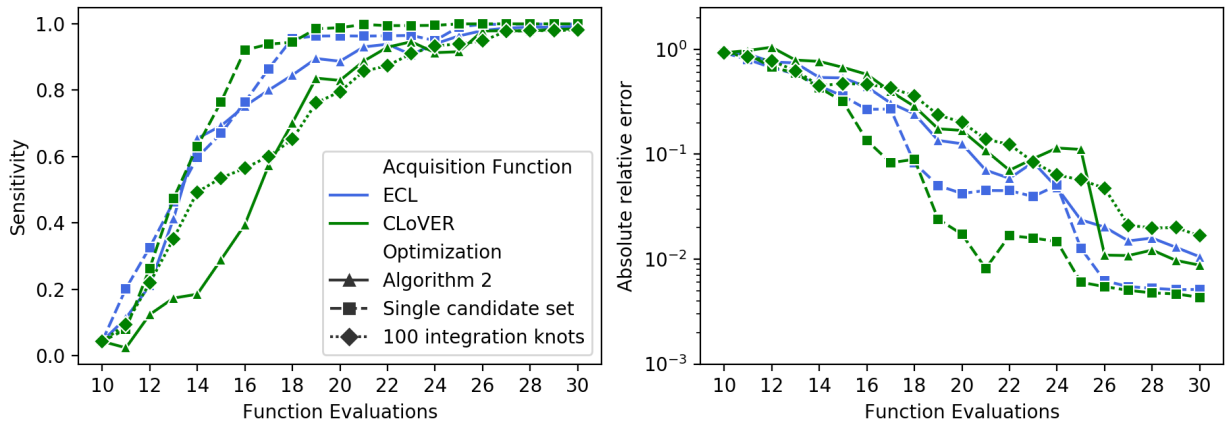


Figure 5.6: Mean sensitivity (left) and relative error of the volume estimate (right) for the Branin-Hoo experiment over 30 MC repetitions. The ECL (blue) and CLoVER (green) acquisition functions are used with different optimization strategies.

the sensitivity and absolute relative error distributions for the 30 repetitions at the experiment's conclusion. Since the number of sequential samples (170) is a large proportion of the candidate set's size, I introduce a third optimization strategy that generates unique 1000-sized LHSs at each adaptive step (blue boxes). For ECL, using Algorithm 6 for optimization (purple box) clearly beats both a single candidate set (yellow box) or unique candidate sets for each iteration.

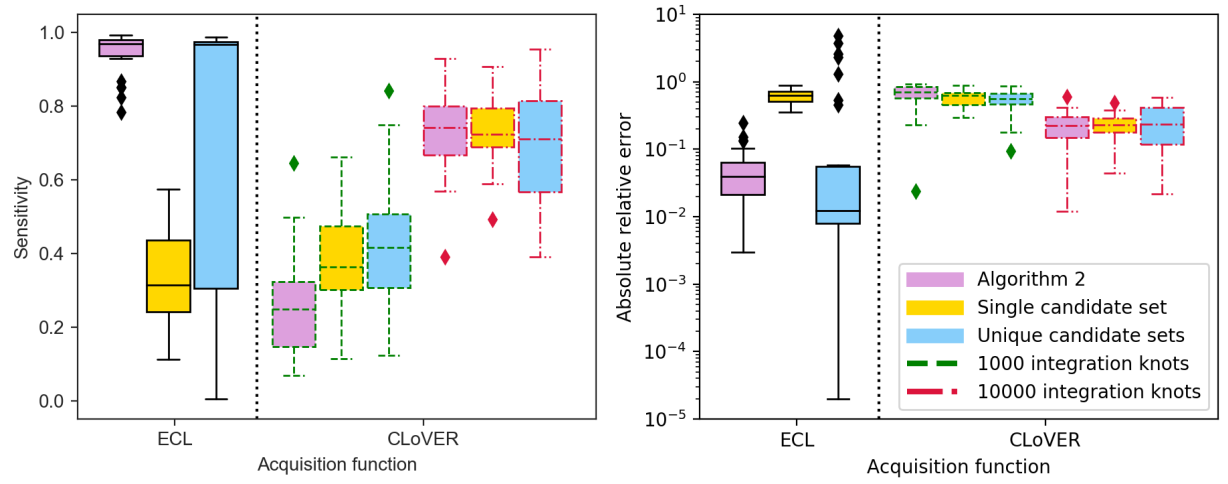


Figure 5.7: Final sensitivity (left) and relative error (right) for the Ishigami experiment. Different optimization strategies are used on the ECL and/or CLoVER acquisition functions, changing the number of candidate points and integration knots.

Unlike with ECL, the CLoVER results in Figure 5.7 do not show a drastic change in sensitivity or relative error across optimization strategies. With the Ishigami experiment being harder (larger quantile, more failure regions) than Branin-Hoo, the original set of 1000 integration knots may be too small. Increasing to 10000 knots shows a significant improvement for all optimization strategies. Looking back at the Branin-Hoo experiment in Figure 5.6, I added a CLoVER variation using a single candidate set and only 100 integration points (dotted line with triangles). I find the sensitivity is below CLoVER with 1000 knots for a majority of the experiment and the relative error is significantly higher.

Optimization Strategy	**ECL**	**CLoVER**	
		10^3 knots	10^4 knots
Algorithm 6	0.008	15.2	17.1
Single candidate set	0.09	17.8	32.4
Unique candidate sets	0.14	18.2	28.7

Table 5.3: Average computation times (minutes) to select points across 30 MC repetitions of the Ishigami experiment.

Increasing the number of optimization candidate points or integration knots comes at a price – namely computation time. Table 5.3 summarizes the drastic difference in compute times between ECL and CLoVER for the Ishigami experiment. For challenging problems like Ishigami, an even greater number of integration knots seems necessary to produce competitively accurate results. When accounting for the additional computational burden, implementing CLoVER with a large number of knots in higher dimensional problems (i.e. Hartmann-6) could be impractical based on current software.

5.5 Failure probability estimation

Here I transition from contour finding to failure probability estimation, combining my ECL adaptive design with MFIS as in Figure 5.1. I refer back to the synthetic functions of Section

5.4 before applying the methodology to my motivating spacesuit impact simulator.

5.5.1 Synthetic benchmarking

Figure 5.5 illustrates how sequential design can furnish accurate GP mean predictions (high sensitivity and low relative error) in the vicinity of a failure region. As introduced in Section 5.2.1 and foreshadowed in Figure 5.2, an unbiased estimate of this volume (i.e., the failure probability) may be improved through further sampling with MFIS. Here I build upon Section 5.4 experiments with the Ishigami and Hartmann-6 functions. With GPs $\mathcal{S}^{\text{Ish}}$ and $\mathcal{S}^{\text{Hart}}$ trained to the ECL adaptive designs as surrogates, or ordinary LHSs as a benchmark, I use my `Python` package `MFISPy`[3] to generate my bias distributions and calculate $\hat{\alpha}$. The `scipy.stats` package was used for the input distributions.

Ishigami used input distribution $\mathbb{F}^{\text{Ish}}$ with independent marginals $x_1 \sim \mathcal{N}(-1, 1)$, $x_2 \sim \mathcal{N}(1.5, 1.5^2)$ and $x_3 \sim \text{Uniform}(-\pi, \pi)$, where the Gaussians are truncated to $\mathcal{X} \in [-\pi, \pi]^3$, which puts $\mathbb{F}^{\text{Ish}}$'s center of the mass near to four of the six failure regions.[4] Combining the limit state function in Table 5.1 along with this input distribution results in $\alpha^{\text{Ish}} \approx 1.9 \times 10^{-4}$. For Hartmann-6 I used $\mathbb{F}^{\text{Hart}}$ comprised of independent $x_j \sim \mathcal{N}(0.5, 0.1^2)$ for $j = 1, \ldots, 6$ truncated to $\mathcal{X} = [0, 1]^6$. Here the failure probability, using the threshold in Table 5.1, is $\alpha^{\text{Hart}} \approx 9.96 \times 10^{-6}$.

Following Algorithm 5, I generated $\acute{N}^{\text{Ish}} = 5 \times 10^6$ samples from $\mathbb{F}^{\text{Ish}}$, obtaining predictive evaluations under $\mathcal{S}^{\text{Ish}}$ at those locations. Due to $\alpha^{\text{Hart}} \ll \alpha^{\text{Ish}}$, I produced many more $\acute{N}^{\text{Hart}} = 5 \times 10^7$ samples from $\mathcal{S}^{\text{Hart}}$ in order to obtain enough failures for the bias distribution training, $\mathbb{F}_{\star}^{\text{Hart}}$. I fit each bias distribution using a Gaussian mixture model of up to 10 clusters, determined with cross-validation, using diagonal covariances. Both experiments

[3] https://github.com/nasa/MFISPy

[4] A modification to the original experiment in Section 5.2.2 (where uniform distributions were used).

operated with a total budget of 1000 evaluations of the true model $\mathcal{T}$. Thus, based on the adaptive design budgets were $N = 200$ for Ishigami and 500 for Hartmann-6, $\acute{N}^\star = 800$ and 500 respectively.

In many engineering applications, budget considerations limit data collection to a single experiment. Generating a single estimate $\hat{\alpha}$ motivates a desire to be conservative in the estimate (Auffray et al., 2014; Azzimonti et al., 2020). As discussed by Dubourg and Deheeger (2011), I can leverage the GP's posterior distribution when classifying failure inputs used to fit the bias distribution. Namely, I define $\hat{y}(\mathbf{x})$ in line 4 of Algorithm 5 as $\hat{y}(\mathbf{x}) = \mu_N(\mathbf{x}) + \delta\sigma_N(\mathbf{x})$, with $\mu_N(\mathbf{x}), \sigma_N(\mathbf{x})$ from equations in (2.2). For my problems I use $\delta = 1.645$, making $\hat{y}(\mathbf{x})$ the 95% upper confidence bound (UCB).

Figure 5.8 shows the failure probability estimates $\hat{\alpha}$ for 30 repetitions in the Ishigami (left) and Hartmann-6 (right) experiments following this scheme. In these problems, the mean surfaces of the LHS-designed GPs rarely exceed the failure threshold, classifying a few (if any) samples as failures, leading to an MFIS estimate of zero. Using the UCB for failure classification often produced enough failures to fit a bias distribution. There is considerable variability in the MFIS estimates based on UCB, but for Ishigami $\hat{\alpha}^{\text{Ish}}$ is within an order of magnitude of the true α^{Ish} (red horizontal dashed line).

Just as in Figure 5.2, using an adaptive design for the GP surrogate $\mathcal{S}$ greatly improves the downstream accuracy of MFIS estimates. Those estimates are centered around the true values (red dashed lines), exemplifying unbiasedness. For the ECL results in Figure 5.2, using the UCB does not help much, in large part due to the much lower uncertainty around the contour thanks to the contour targeting in adaptive design. Yet in cases where significant uncertainty remains after the adaptive design (LHS results), using UCB may result in a conservative bias distribution over undiscovered sections of the failure region(s).

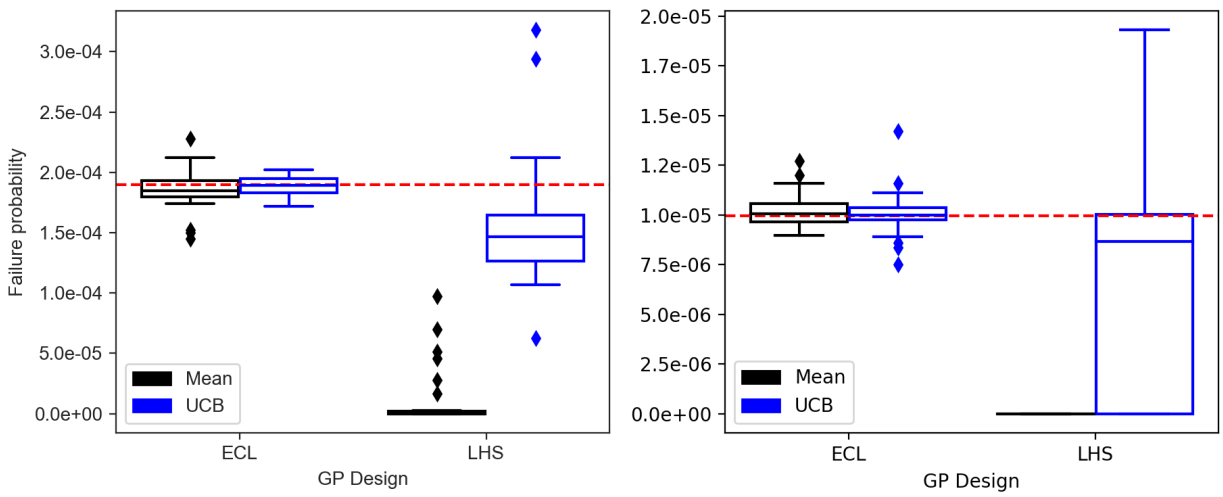


Figure 5.8: MFIS failure probability estimates $\hat{\alpha}$ for the Ishigami (left) and Hartmann-6 (right) based on GPs trained from ECL adaptive and LHS designs over 30 MC repetitions. The color of each box shows the definition of $\hat{y}(\mathbf{x})$ (black: GP mean; blue: GP 95% upper confidence bound) used to classify failure locations for the bias distribution $\mathbb{F}_\star$. The true failure probabilities α are shown with the red-dashed lines.

5.5.2 Spacesuit impact simulation

NASA's Artemis program is centered around a renewed effort of manned space exploration. Within this program, NASA is working to develop the next-generation spacesuit, the exploration extravehicular mobility unit (xEMU), to provide astronauts with a suit that is robust, lightweight, and mobile. Under potential impact loads (due to projectiles, falls, etc.) the suit's probability of no impact failure (PnIF) is of top concern for certification. With the computational model for the xEMU currently under development, I implemented my methodology on a computer model of the previous generation spacesuit, `Z-2`, to estimate PnIF under various impact loading conditions. The model simulates an impact to the hard upper torso region of the spacesuit (shown in Figure 5.9) using LS-Dyna finite element method software (Gladman et al., 2007) and the MAT162 material model (Gama et al., 2009; Haque, 2017) to capture the resulting progressive damage to the material. Based on calibration and sensitivity analysis detailed in Warner et al. (2021), there are four variables

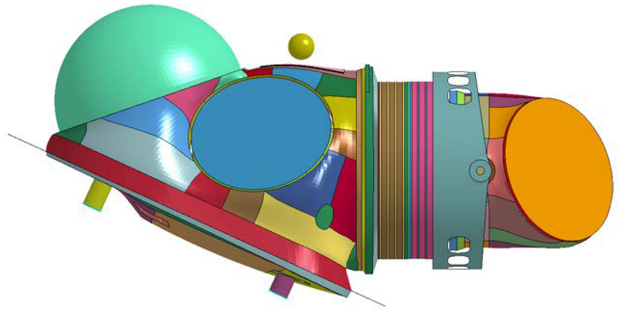


Figure 5.9: MAT162 assembly model for hard upper torso of `Z-2` spacesuit and a rock projectile (yellow sphere).

in my experiment: three parameters controlling the softening behavior of the material due to damage, henceforth referred to as damage coefficients 1-3, and the impact velocity. The input distribution $\mathbb{F}$ is $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where

$$\boldsymbol{\mu} = \begin{bmatrix} 0.41597 \\ 1.54189 \\ 0.01031 \\ 1 \end{bmatrix} \text{ and } \boldsymbol{\Sigma} = \begin{bmatrix} 0.00275 & -0.00494 & -0.00373 & 0 \\ -0.00494 & 0.01856 & 0.0032 & 0 \\ -0.00373 & 0.0032 & 0.01834 & 0 \\ 0 & 0 & 0 & 0.00016 \end{bmatrix}. \tag{5.8}$$

The response of interest, y, is the maximum contact force, with a threshold of 2800 lbf. Thus to estimate $\alpha = 1 - \text{PnIF}$, I define $g(y(\mathbf{x})) = y(\mathbf{x}) - 2800$. The computer model requires 18 hours on 10 cores to produce one sample, severely limiting my ability to entertain a large campaign. I budgeted $N = 200$ runs for the adaptive design (including an initial LHS of 40), with adaptive samples generated in batches of ten so the required simulations could be performed in parallel. For the GP surrogate $\mathcal{S}$, I use the Matérn 3/2 (Stein, 2012) kernel with separable lengthscale and a jitter of 10^{-6}. The data is prescaled to $[0, 1]^4$ using bounds derived from five standard deviations away from the mean in each dimension.

As a benchmarking comparator another GP was fit to a 200-sized LHS that was warped to $\mathbb{F}$ in equation (5.8) via an inverse CDF (dropping the covariances). Figure 5.10 shows

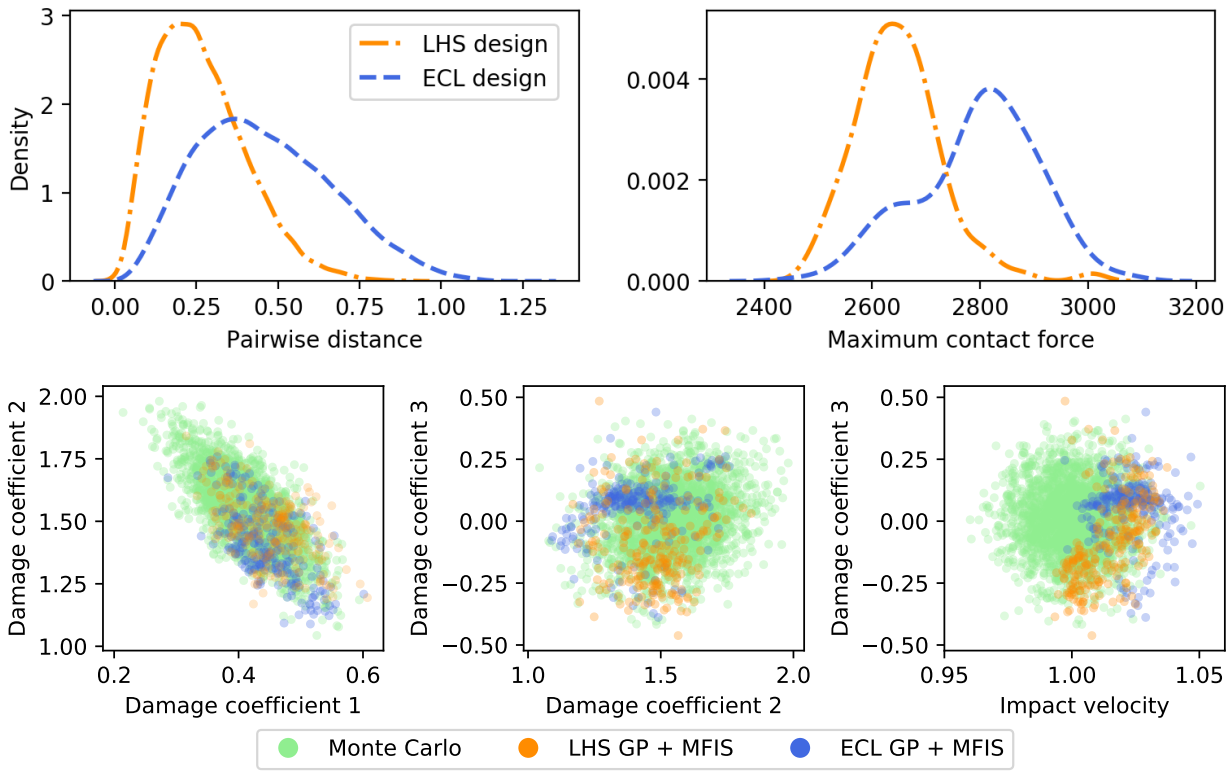


Figure 5.10: GP designs $(\mathbf{X}_N, \mathbf{Y}_N)$ and MFIS samples $\mathbf{X}_{\acute{N}^\star}$ for the Z−2 spacesuit experiment. *Top left:* pairwise distances for the inputs. *Top right:* observed outputs from GP designs. *Bottom:* pairwise plots of samples from input ($\mathbb{F}$, green) and bias ($\mathbb{F}_\star$, orange and blue) distributions used for estimating failure probability.

pairwise observed distances in both designs' set of inputs. The ECL design contains more long distances between points – the inputs are more spread out in the input space. The density in the top right of Figure 5.10 shows the focus of ECL's samples around $T = 2800$. I take these distinctions as circumstantial evidence that my sequential design method is working.

I used $\acute{N} = 10^5$ samples and the $\mathcal{S}$'s UCB to train the bias distribution $\mathbb{F}_\star$. Due to the covariances (5.8) between inputs in $\mathbb{F}$, a Gaussian mixture model with a full covariance structure was used for $\mathbb{F}_\star$. I generated $M^\star = 250$ samples from both bias distributions in order to calculate MFIS estimates. The plots in the bottom of Figure 5.10 show projections

of these samples for select pairs of input variables. While the densities of points look similar for damage coefficient 2 compared to damage coefficient 1, the ECL MFIS samples are biased higher for damage coefficient 3 and impact velocity than compared to LHS MFIS samples. This confirms intuition that higher impact velocity produces more damage, and thus produces failures. As a reference, a set of $M = 2500$ pure-MC samples is also shown (generated from $\mathbb{F}$). Both sets of samples from bias distributions show a clear tendency towards sections of the input space targeting predicted failure region(s).

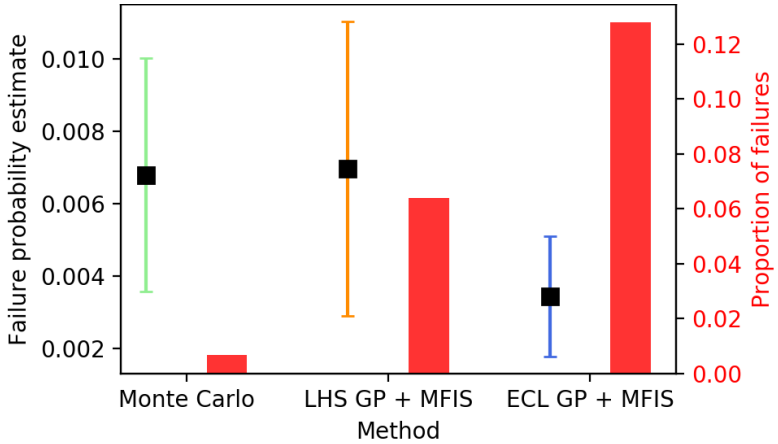


Figure 5.11: Failure probability estimates $\hat{\alpha}$ with 95% confidence intervals. Red bars show the proportion of samples with failure outputs that were used in those estimates.

MFIS estimates based on the samples in Figure 5.10 are shown in Figure 5.11. Notice that the 95% confidence intervals of all three estimates overlap, with the ECL GP + MFIS giving the smallest point estimate. The ECL GP + MFIS estimate also contains the narrowest confidence interval, due to a higher proportion of failures observed in the MFIS samples. The proportion of failures observed in both MFIS estimates may seem low and are no doubt affected by using the GP UCB for classification during $\mathbb{F}_\star$ fitting. A higher failure rate for ECL GP + MFIS samples suggests that the bias distribution produced by the UCB of the ECL GP includes the unknown true failure region as a larger proportion of the bias distribution's density.

5.6 Discussion

Estimating reliability in engineering applications often entails expensive simulation, limiting the ability to gather large quantities of evaluations. Multifidelity importance sampling is one strategy to reduce the amount of data needed for an unbiased reliability estimate, leveraging a surrogate model (like a GP) and bias distribution to reduce estimator variance. If the GP's predictive surface is inaccurate in the vicinity of the true failure contour, misclassified samples may be used to fit a bias distribution, threatening the overall accuracy of the reliability estimate. By marrying a GP built on an adaptive design for contour location with MFIS, I can guard against such pitfalls.

I proposed a thrifty adaptive design with a novel criteria: the Entropy-based Contour Locator (ECL). My ECL acquisition function applies the basic definition of entropy on events derived from predicted pass/fail from the surrogate. Everything for ECL has a closed form. I solve for a new acquisition, maximizing ECL, in such a way that multiple failure regions based on high quantiles may be identified. Results showed that my ECL adaptive designs exhibited a degree of stochasticity in optimization that promoted exploration of local maxima. When problems contain multiple failure regions, especially ones with small relative volumes, this exploration is key. my examples include illustrative benchmarks with two (2d and 6d) and six (3d) failure regions. Against a litany of contemporary adaptive design benchmarks, ECL performs among the best at correctly identifying true failures as well as estimating the contour's volume. All this is accomplished at a one to two orders of magnitude computational savings. I also implemented ECL design plus MFIS on spacesuit damage simulator, demonstrating how combining these two methods provides a point estimate with less uncertainty than plain MC or MFIS techniques.

I demonstrated how the adaptive design process with ECL may be extended to batch

selection, providing computational savings when parallel-processing is available. By selecting multiple points before observing more responses, the batch option is more sensitive to the quality of surrogate, and in particular its contour estimate(s). Batch size also plays a role in the quality of the information gained from the sample points.

My methodological contribution focused on targeting failure regions with ECL, not explicitly on its interface with MFIS. Rather, I treated MFIS as one, of possibly several, downstream reliability tasks after ECL acquisition. This compartmental nature is an asset, as it means my contribution is portable. However, it also means potential for improved efficiency in the reliability context was left on the table. I believe there is opportunity for further blending of adaptive design and MFIS: perhaps by generating acquisitions and refining MFIS bias distributions $\mathbb{F}_\star$ at the same time.

Chapter 6

Final thoughts

6.1 Conclusion

Advancements in data collection and simulations make today an exciting time to be a modeler and statistician. Even when the opportunity to collect sample data is limited, computer experiments serve as a crucial alternative to traditional physical experiments. Gaussian proccesses (GPs) act as flexible models that provide accurate predictions and uncertainty estimates for such experiments. GPs also provide efficient strategies for conducting and analyzing computer experiments.

The size of data sets continues to grow exponentially, stressing a GP's ability to leverage all the information for modeling and prediction. In Chapter 3, I introduce the locally induced GP (LIGP), which marries the idea of building local approximate GPs (LAGPs) with inducing points to reduce the computational burden of building a GP for a large-scale determinishic simulation. With this new set of local approximate models, I develop the weighted integrated mean-squared error (wIMSE) criterion to select the inducing points. The focus of selecting inducing point was extended to a template (including space-filling alternatives), reusing the same design for each local GP. My results show that inducing points reduce computational cost for model building/prediction, allowing for larger neighborhoods in local models and comparable (sometimes superior) predictive accuracy. I upgrade LIGP to model stochastic simulations in Chapter 4. When replicates exist in the data, which is

often needed to separate signal from noise, I use the Woodbury identities to build neighborhoods and perform inference and prediction based on the number of unique design locations. Through synthetic experiments, I show that LIGP provides the opportunity to include a neighborhood that is a magnitude larger than LAGP with more accurate noise estimates and predictions.

Chapter 5 addresses a class of problems when data is scarce due to an expensive simulator. Leveraging a GP's adaptive design ability, I introduce a novel entropy-based contour locator (ECL) function to conduct an experiment focused on learning a contour. In particular, my methodology and implementation is motivated by problems with extremely low/high contour and multiple contour regions. I show that using ECL with a simple optimization strategy meets or exceeds the ability of other contour finding acquisition functions to construct a design that correctly identifies points in the contour and the contour's volume. Using a GP based on an ECL adaptive design within multifidelity importance sampling produces an unbiased estimate of the contour volume (i.e. failure probability) with lower uncertainty. This is showcased on NASA's `Z-2` spacesuit impact simulator.

6.2 Future directions

6.2.1 Extensions for locally induced Gaussian processes

With many successes in LIGP in Chapters 3 and 4, there are still opportunities to improve LIGP implementation. The Matérn kernel is currently not featured in the LIGP code, but would provide a more flexible covariance structure for a bumpier mean surface. In addition, modeling covariance in high dimensions can be helped by extending lengthscale estimation to a separable (anisotropic) structure. LIGP could also benefit from neighborhood built with

a variance metric, such as ALC or its thriftier alternative `alcray` in `laGP` (Gramacy and Haaland, 2016). While each of these would require more computation, they each have the potential to minimize that impact with superior predictive accuracy.

Another feature not fully explored is how to determine an appropriate neighborhood size and the number of inducing points to model a given data set. The data's dimensionality, signal-to-noise ratio, and smoothness of the mean surface likely all play a role in the selection $n/\bar{n}$ and m. While a simple experiment on the values of m, n was conducted in Section 3.4.3, this is not practical in most problems. A more plug-n-play strategy to determine m, n without wasting excessive computing resources remains elusive.

At this time, local models like LAGP and LIGP are built strictly for independent pointwise-prediction. It would be interesting to consider extending a single LIGP model for prediction at a set of locations, which could be completed with relying upon the reference set in ALC design. The local neighborhood itself could be constructed around a similar reference set. If the set of predictions is along a distribution, then the weight in wIMSE could be adjusted for inducing point selection. Examples exist in finance applications where a set of predictions with an associated probability density is needed to estimate an expectation.

6.2.2 Contour targeting and reliability

The encouraging results of ECL design inspire further directions for research. Performing ECL design in batches shows great potential for saving wall-clock time. Yet there are likely extremes for the batch size at which point designs with the same number of data points produce very different contour estimates. I believe a thorough investigation is needed of the limits of batch sizes versus problem specifications such as design size, number of dimensions, and quantile. One might be able to borrow from the strategy multifidelity methods (Marques

et al., 2018), which seeks to weight a sample's output source with the cost (i.e. computation time) associated with each fidelity model. In the case of ECL, a recommended batch size could come from considering the cost of the high-fidelity model and potential for time savings.

In Chapter 5, all simulation models (and synthetic functions) produced deterministic output. Recently the goal of contour finding is being extended to stochastic (even heteroskedastic) simulators (Lyu et al., 2018). It would be helpful to extend ECL's design and modeling for noisy outputs and promote replication for distinguishing signal from noise. The ability to conduct ECL design is batches with some replication increases potential for larger batch sizes.

Section 5.5.2 demonstrates the potential for ECL design to drive down uncertainty in reliability estimates through contour targeting. Other similar applications provide opportunities for fine tuning the algorithm to handle a more challenging set of input variables. The initial space-filling design and ECL optimization is currently based on a finite rectangular design region. However in manufacturing and engineering applications, it is common to encounter non-rectangular design regions (Nguyen and Piepel, 2005). Discrete and/or categorical inputs are also possible, which has been addressed in the Bayesian optimization setting (Pelamatti et al., 2021; Zhang et al., 2020b). To extend ECL adaptive design to more applications, a more flexible framework is warranted that includes modeling an irregular design region with a mixture of continuous and discrete/categorical inputs. In the spacesuit simulator example, the input distribution defined a non-finite design region, which I truncated. I believe that further exploration on how to handle adaptive design in non-finite design regions, such as in Ha et al. (2019), would be of value.

In practice there may be situations where the failure threshold is not a fixed, known constant. This could be due to uncertainty in the classification or material degradation. Engineers may seek to estimate reliability with an uncertainty distribution on the threshold.

Similar work exists for reliability and degradation modeling (Usynin et al., 2008; Wang and Coit, 2007; Zhu et al., 2017). Incorporating a similar capability in ECL design may further extend its relevance in solving modern problems.

Another future direction is improving the MFIS estimate for reliability calculations. The current ECL design in Algorithm 6 is largely disjoint from the MFIS Algorithm 5. But since the GP is being updated sequentially, one wonders if the same can be done for the bias distribution. By tracking changes in the bias distribution, it could help determine when enough of the overall function evaluation budget has been spent on the adaptive design. Further work may yield an opportunity to include the adaptive samples in the final MFIS estimate through a similar re-weighting scheme. In addition, there may be a way to incorporate field data with the simulated data in either stage of the estimation process. Similar work in multistage Markov Chain Monte Carlo (Christen and Fox, 2005; Fox and Nicholls, 1997) may be applied to drive field data sampling in areas of high uncertainty or predicted failure regions.

Bibliography

Petter Abrahamsen. A review of gaussian random fields and correlation functions, 1997.

Christoforos Anagnostopoulos and Robert B Gramacy. Information-theoretic data discarding for dynamic trees on data streams. *Entropy*, 15(12):5510–5535, 2013.

Bruce Ankenman, Barry L Nelson, and Jeremy Staum. Stochastic kriging for simulation metamodeling. *Operations research*, 58(2):371–382, 2010.

Siu-Kui Au and James L Beck. Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic engineering mechanics*, 16(4):263–277, 2001.

Yves Auffray, Pierre Barbillon, and Jean-Michel Marin. Bounding rare event probabilities in computer experiments. *Computational Statistics & Data Analysis*, 80:153–166, 2014.

Erlend Aune, Daniel P Simpson, and Jo Eidsvik. Parameter estimation in high dimensional gaussian distributions. *Statistics and Computing*, 24(2):247–263, 2014.

Dario Azzimonti, Julien Bect, Clément Chevalier, and David Ginsbourger. Quantifying uncertainties on excursion sets under a gaussian random field prior. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):850–874, 2016.

Dario Azzimonti, David Ginsbourger, Clément Chevalier, Julien Bect, and Yann Richet. Adaptive design of experiments for conservative estimation of excursion sets. *Technometrics*, pages 1–14, 2020.

Shan Ba, William R Myers, and William A Brenneman. Optimal sliced latin hypercube designs. *Technometrics*, 57(4):479–487, 2015.

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. *arXiv preprint arXiv:2002.01321*, 2020.

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.

S. Barnett. *Matrix Methods for Engineers and Scientists*. McGraw-Hill, 1979.

Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse gaussian process approximations. *Advances in Neural Information Processing Systems*, 29:1533–1541, 2016.

Julien Bect, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez. Sequential design of computer experiments for the estimation of a probability of failure. *Statistics and Computing*, 22(3):773–793, 2012.

Julien Bect, François Bachoc, and David Ginsbourger. A supermartingale approach to Gaussian process based sequential design of experiments. *Preprint on arXiv:1608.01118*, 2016.

Julien Bect, Ling Li, and Emmanuel Vazquez. Bayesian subset simulation. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):762–786, 2017.

Barron J Bichon, Michael S Eldred, Laura Painton Swiler, Sandaran Mahadevan, and John M McFarland. Efficient global reliability analysis for nonlinear implicit performance functions. *AIAA journal*, 46(10):2459–2468, 2008.

Mickael Binois and Robert B Gramacy. *`hetGP`: Heteroskedastic Gaussian Process Modeling and Design under Replication*, 2018.

Mickaël Binois, Robert B Gramacy, and Mike Ludkovski. Practical heteroskedastic Gaussian process modeling for large simulation experiments. *Journal of Computational and Graphical Statistics*, 27(4):808–821, 2018a.

Mickaël Binois, Jiangeng Huang, Robert B Gramacy, and Mike Ludkovski. Replication or exploration? Sequential design for stochastic simulation experiments. *Technometrics*, 61 (1):7–23, 2018b.

Ilija Bogunovic, Jonathan Scarlett, Andreas Krause, and Volkan Cevher. Truncated variance reduction: A unified approach to bayesian optimization and level-set estimation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 1507–1515. Curran Associates, Inc., 2016.

David Bolin and Finn Lindgren. Excursion and contour uncertainty regions for latent gaussian models. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, pages 85–106, 2015.

Andrew J Booker, John E Dennis, Paul D Frank, David B Serafini, Virginia Torczon, and Michael W Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization*, 17(1):1–13, 1999.

George EP Box and Norman R Draper. *Empirical model-building and response surfaces.* John Wiley & Sons, 1987.

George EP Box and Norman R Draper. *Response surfaces, mixtures, and ridge analyses*, volume 649. John Wiley & Sons, 2007.

Brent Bryan and Jeff Schneider. Actively learning level-sets of composite functions. In *Proceedings of the 25th international conference on Machine learning*, pages 80–87, 2008.

Evgeny Burnaev and Maxim Panov. Adaptive design of experiments based on gaussian processes. In *International Symposium on Statistical Learning and Data Sciences*, pages 116–125. Springer, 2015.

Daniel Busby. Hierarchical adaptive experimental design for gaussian process emulators. *Reliability Engineering & System Safety*, 94(7):1183–1193, 2009.

Richard H Byrd, Peihuang Qiu, Jorge Nocedal, , and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *Journal on Scientific Computing*, 16(5):1190–1208, 1995.

Claire Cannamela, Josselin Garnier, Bertrand Iooss, et al. Controlled stratification for quantile estimation. *The Annals of Applied Statistics*, 2(4):1554–1580, 2008.

Olivier Cappé, Arnaud Guillin, Jean-Michel Marin, and Christian P Robert. Population monte carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929, 2004.

Olivier Cappé, Randal Douc, Arnaud Guillin, Jean-Michel Marin, and Christian P Robert. Adaptive importance sampling in general mixture classes. *Statistics and Computing*, 18 (4):447–459, 2008.

Rob Carnell. *lhs: Latin Hypercube Samples*, 2019. URL https://CRAN.R-project.org/package=lhs. R package version 1.0.1.

Gilles Celeux and Gérard Govaert. Gaussian parsimonious clustering models. *Pattern recognition*, 28(5):781–793, 1995.

Jie Chen, Nannan Cao, Kian Hsiang Low, Ruofei Ouyang, Colin Keng-Yan Tan, and Patrick Jaillet. Parallel gaussian process regression with low-rank covariance matrix approximations. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 152–161, Arlington, Virginia, USA, 2013. AUAI Press.

Clément Chevalier, David Ginsbourger, Julien Bect, and Ilya Molchanov. Estimating and quantifying uncertainties on level sets using the vorob'ev expectation and deviation with gaussian process models. In *mODa 10–Advances in Model-Oriented Design and Analysis*, pages 35–43. Springer, 2013.

Clément Chevalier, Julien Bect, David Ginsbourger, Emmanuel Vazquez, Victor Picheny, and Yann Richet. Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. *Technometrics*, 56(4):455–465, 2014.

J Andrés Christen and Colin Fox. Markov chain monte carlo using an approximation. *Journal of Computational and Graphical statistics*, 14(4):795–810, 2005.

K Andrew Cliffe, Mike B Giles, Robert Scheichl, and Aretha L Teckentrup. Multilevel monte carlo methods and applications to elliptic pdes with random coefficients. *Computing and Visualization in Science*, 14(1):3, 2011.

David A. Cohn. Neural network exploration using optimal experiment design. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 679–686. Morgan-Kaufmann, 1994.

Thomas M Cover. *Elements of information theory.* John Wiley & Sons, 2006.

Noel Cressie. *Statistics for spatial data.* John Wiley & Sons, 2015.

Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14 (3):641–668, 2002.

Carla Currin, Toby Mitchell, Max Morris, and Don Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

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.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Kun Dong, David Eriksson, Hannes Nickisch, David Bindel, and Andrew G Wilson. Scalable log determinants for gaussian process kernel learning. In *Advances in Neural Information Processing Systems*, pages 6327–6337, 2017.

Randal Douc, Arnaud Guillin, J-M Marin, and Christian P Robert. Minimum variance importance sampling via population monte carlo. *ESAIM: Probability and Statistics*, 11: 427–447, 2007a.

Randal Douc, Arnaud Guillin, J-M Marin, Christian P Robert, et al. Convergence of adaptive mixtures of importance sampling schemes. *The Annals of Statistics*, 35(1):420–448, 2007b.

V Dubourg and F Deheeger. Metamodel-based importance sampling for the simulation. *Applications of Statistics and Probability in Civil Engineering*, 26:192, 2011.

Delphine Dupuy, Céline Helbert, and Jessica Franco. DiceDesign and DiceEval: Two R packages for design and analysis of computer experiments. *Journal of Statistical Software*, 65(11):1–38, 2015. URL http://www.jstatsoft.org/v65/i11/.

Benjamin Echard, Nicolas Gayton, and Maurice Lemaire. Kriging-based monte carlo simulation to compute the probability of failure efficiently: Ak-mcs method. *6emes Journées Nationales de Fiabilité, 24–26 mars, Toulouse, France*, 2010.

Dirk Eddelbuettel. *Seamless* R *and* C++ *integration with* `Rcpp`. Springer, New York, 2013. ISBN 978-1-4614-6867-7.

Adam M Edwards and Robert B Gramacy. Precision aggregated local models. *arXiv preprint arXiv:2005.13375*, 2020.

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

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.

Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and modeling for computer experiments*. CRC press, 2005.

Fernando Llorente Fernández, Luca Martino, Victor Elvira, David Delgado, and Javier López-Santiago. Adaptive quadrature schemes for bayesian inference via active learning. *IEEE Access*, 8:208462–208483, 2020.

M Giselle Fernández-Godino, Chanyoung Park, Nam-Ho Kim, and Raphael T Haftka. Review of multi-fidelity models. *arXiv preprint arXiv:1609.07196*, 2016.

M Giselle Fernández-Godino, Chanyoung Park, Nam H Kim, and Raphael T Haftka. Issues in deciding whether to use multifidelity surrogates. *AIAA Journal*, 57(5):2039–2054, 2019.

Andrew O Finley, Huiyan Sang, Sudipto Banerjee, and Alan E Gelfand. Improving the performance of predictive process modeling for large datasets. *Computational statistics & data analysis*, 53(8):2873–2884, 2009.

Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

Colin Fox and Geoff Nicholls. Sampling conductivity images via mcmc. *The art and science of Bayesian image analysis*, pages 91–100, 1997.

Bazle A Gama, Travis A Bogetti, and John W Gillespie Jr. Progressive damage modeling of plain-weave composites using ls-dyna composite damage model mat162. In *7th European LS-DYNA Conference*, pages 14–15, 2009.

Jacob R Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018a.

Jacob R Gardner, Geoff Pleiss, Ruihan Wu, Kilian Q Weinberger, and Andrew Gordon Wilson. Product kernel interpolation for scalable gaussian processes. *arXiv preprint arXiv:1802.08903*, 2018b.

Nathaniel Garton, Jarad Niemi, and Alicia Carriquiry. Knot selection in sparse gaussian processes. *arXiv preprint arXiv:2002.09538*, 2020.

Bertrand Gauthier and Luc Pronzato. Spectral approximation of the imse criterion for optimal designs in kernel-based interpolation models. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):805–825, 2014.

Marc G Genton. Classes of kernels for machine learning: a statistics perspective. *Journal of machine learning research*, 2(Dec):299–312, 2001.

Alan Genz and Frank Bretz. *Computation of multivariate normal and t probabilities*, volume 195. Springer Science & Business Media, 2009.

Alan Genz, Frank Bretz, Tetsuhisa Miwa, Xuefei Mi, Friedrich Leisch, Fabian Scheipl, and Torsten Hothorn. `mvtnorm`*: Multivariate normal and t distributions*, 2019. URL https://CRAN.R-project.org/package=mvtnorm. R package version 1.0-10.

Michael B Giles. Multilevel monte carlo path simulation. *Operations research*, 56(3):607–617, 2008.

David Ginsbourger and Rodolphe Le Riche. Towards gaussian process-based optimization with finite time horizon. In *mODa 9–Advances in Model-Oriented Design and Analysis*, pages 89–96. Springer, 2010.

Brian Gladman et al. *LS-DYNA keyword users' manual (version 971)*. Livermore Software Technology Corporation, 2007.

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

Paul W Goldberg, Christopher KI Williams, and Christopher M Bishop. Regression with input-dependent noise: a gaussian process treatment. *Advances in neural information processing systems*, 10:493–499, 1997.

Javier González, Michael Osborne, and Neil Lawrence. Glasses: Relieving the myopia of bayesian optimisation. In *Artificial Intelligence and Statistics*, pages 790–799, 2016.

Alex Gorodetsky and Youssef Marzouk. Mercer kernels and integrated variance experimental design: connections between gaussian process regression and polynomial approximation. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):796–828, 2016.

Alkis Gotovos. Active learning for level set estimation. Master's thesis, Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 2013.

Robert B Gramacy. "optimization under unknown constraints," in proceedings of the ninth valencia international meetings on bayesian statistics, 2011.

Robert B Gramacy. `laGP`: Large-scale spatial modeling via local approximate Gaussian processes in `r`. *Journal of Statistical Software*, 72(1):1–46, 2016.

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/.

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.

Robert B Gramacy and Benjamin Haaland. Speeding up neighborhood search in local Gaussian process prediction. *Technometrics*, 58(3):294–303, 2016.

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. ISSN 01621459.

Robert B Gramacy and Herbert KH Lee. Adaptive design and analysis of supercomputer experiments. *Technometrics*, 51(2):130–145, 2009.

Robert B Gramacy and Herbert KH Lee. Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22(3):713–722, 2012.

Robert B Gramacy and Nicholas G Polson. Particle learning of Gaussian process models for sequential design and optimization. *Journal of Computational and Graphical Statistics*, 20(1):102–118, 2011.

Robert B Gramacy, Jarad Niemi, and Robin M Weiss. Massively parallel approximate Gaussian process regression. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1): 564–584, 2014.

Robert B Gramacy, Derek Bingham, James Paul Holloway, Michael J Grosskopf, Carolyn C Kuranz, Erica Rutter, Matt Trantham, R Paul Drake, et al. Calibrating a large computer experiment simulating radiative shock hydrodynamics. *The Annals of Applied Statistics*, 9(3):1141–1168, 2015.

Robert B Gramacy, Genetha A Gray, Sébastien Le Digabel, Herbert KH Lee, Pritam Ranjan, Garth Wells, and Stefan M Wild. Modeling an augmented lagrangian for blackbox constrained optimization. *Technometrics*, 58(1):1–11, 2016.

Huong Ha, Santu Rana, Sunil Gupta, Thanh Nguyen, Hung Tran-The, and Svetha Venkatesh. Bayesian optimization with unknown search space. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Bazle Z Gama Haque. A progressive composite damage model for unidirectional and woven fabric composites. *Materials Science Corporation (MSC) and University of Delaware Center for Composite Materials (UD-CCM)*, 2017.

David A Harville. Matrix algebra from a statistician's perspective, 1998.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

Radu Herbei and L Mark Berliner. Estimating ocean circulation: an mcmc approach with approximated likelihoods via the bernoulli factory. *Journal of the American Statistical Association*, 109(507):944–954, 2014.

William Herlands, Andrew Wilson, Hannes Nickisch, Seth Flaxman, Daniel Neill, Wilbert Van Panhuis, and Eric Xing. Scalable gaussian processes for characterizing multidimensional change surfaces. In *Artificial Intelligence and Statistics*, pages 1013–1021. PMLR, 2016.

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

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.

Trong Nghia Hoang, Quang Minh Hoang, and Bryan Kian Hsiang Low. A unifying framework of anytime sparse gaussian process regression models with stochastic variational inference for big data. In *ICML*, pages 569–578, 2015.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

L.J. Hong and B.L. Nelson. Discrete optimization via simulation using COMPASS. *Operations Research*, 54(1):115–129, 2006.

Ruimeng Hu and Mike Ludkovski. Sequential design for ranking response surfaces. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):212–239, 2017.

Ronald L Iman and WJ Conover. Small sample sensitivity analysis techniques for computer

models. with an application to risk assessment. *Communications in statistics-theory and methods*, 9(17):1749–1842, 1980.

Tsutomu Ishigami and Toshimitsu Homma. An importance quantification technique in uncertainty analysis for computer models. In *[1990] Proceedings. First International Symposium on Uncertainty Modeling and Analysis*, pages 398–403. IEEE, 1990.

Martin Jankowiak and Jacob Gardner. Neural likelihoods for multi-output gaussian processes, 2019.

Leah R Johnson, Robert B Gramacy, Jeremy Cohen, Erin Mordecai, Courtney Murdock, Jason Rohr, Sadie J Ryan, Anna M Stewart-Ibarra, Daniel Weikel, et al. Phenomenological forecasting of disease incidence using heteroskedastic gaussian processes: A dengue case study. *The Annals of Applied Statistics*, 12(1):27–66, 2018.

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.

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.

V Roshan Joseph, Evren Gul, and Shan Ba. Maximum projection designs for computer experiments. *Biometrika*, 102(2):371–380, 2015.

Motonobu Kanagawa and Philipp Hennig. Convergence guarantees for adaptive bayesian quadrature methods. In *Advances in Neural Information Processing Systems*, pages 6237–6248, 2019.

Matthias Katzfuss and Joseph Guinness. A general framework for vecchia approximations of gaussian processes. *Statist. Sci.*, 36(1):124–141, 02 2021.

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

Carl G Kaufman, Derek Bingham, Salman Habib, Katrin Heitmann, and Joshua A 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.

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.

Marc C Kennedy and Anthony O'Hagan. Supplementary details on bayesian calibration of computer models. Technical report, Internal Report. URL http://www. shef. ac. uk/~ st1ao/ps/calsup. ps, 2001.

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, pages 393–400, 2007.

Hyoung Moon Kim, Bani K. Mallick, and C. C. Holmes. Analyzing nonstationary spatial data using piecewise Gaussian processes. *Journal of the American Statistical Association*, 100(470):653–668, 2005. ISSN 01621459.

Remi Lam, Karen Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *Advances in Neural Information Processing Systems*, pages 883–891, 2016.

Erin R Leatherman, Thomas J Santner, and Angela M Dean. Computer experiment designs for accurate prediction. *Statistics and Computing*, 28(4):739–751, 2018.

Herbert KH Lee, Robert B Gramacy, Crystal Linkletter, and Genetha A Gray. Optimization subject to hidden constraints via statistical emulation. *Pacific Journal of Optimization*, 7(3):467–478, 2011.

Tae H Lee and JJ Jung. A sampling technique enhancing accuracy and efficiency of metamodel-based rbdo: Constraint boundary sampling. *Computers & Structures*, 86(13): 1463–1476, 2008. ISSN 0045-7949. Structural Optimization.

Benjamin Letham and Eytan Bakshy. Bayesian optimization for policy search via online-offline experimentation. *Journal of Machine Learning Research*, 20(145):1–30, 2019.

Benjamin Letham, Brian Karrer, Guilherme Ottoni, Eytan Bakshy, et al. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

Jing Li and Dongbin Xiu. Evaluation of failure probability via surrogate models. *Journal of Computational Physics*, 229(23):8966–8980, 2010.

Jing Li, Jinglai Li, and Dongbin Xiu. An efficient surrogate-based method for computing rare failure probability. *Journal of Computational Physics*, 230(24):8683–8697, 2011.

Alexander Litvinenko, Hermann G Matthies, and Tarek A El-Moselhy. Sampling and low-rank tensor approximation of the response surface. In *Monte Carlo and Quasi-Monte Carlo Methods 2012*, pages 535–551. Springer, 2013.

Haitao Liu, Jianfei Cai, Yew-Soon Ong, and Yi Wang. Understanding and comparing scalable gaussian process regression for big data. *Knowledge-Based Systems*, 164:324–335, 2019.

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

Xiong Lyu, Mickael Binois, and Michael Ludkovski. Evaluating gaussian process metamodels and sequential designs for noisy level set estimation. *arXiv preprint arXiv:1807.06712*, 2018.

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

Alexandre Marques, Remi Lam, and Karen Willcox. Contour location via entropy reduction leveraging multiple information sources. In *Advances in neural information processing systems*, pages 5217–5227, 2018.

Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

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.

Ian W McKeague, Geoff Nicholls, Kevin Speer, and Radu Herbei. Statistical inversion of south atlantic circulation in an abyssal neutral density layer. *Journal of Marine Research*, 63(4):683–704, 2005.

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.

Robert E Melchers and André T Beck. *Structural reliability analysis and prediction*. John Wiley & Sons, 2018.

Toby J Mitchell and David S Scott. A computer program for the design of group testing experiments. *Communications in Statistics-Theory and methods*, 16(10):2943–2955, 1987.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

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

Raymond H Myers, Douglas C Montgomery, and Christine M Anderson-Cook. *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons, 2016.

Radford M Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification. *arXiv preprint physics/9701026*, 1997.

Radford M Neal. Regression and classification using Gaussian process priors. *Bayesian Statistics*, 6:475–501, 1998.

Nam-Ky Nguyen and Greg F Piepel. Computer-generated experimental designs for irregular-shaped regions. *Quality Technology & Quantitative Management*, 2(2):147–160, 2005.

Thomas Nickson, Tom Gunter, Chris Lloyd, Michael A Osborne, and Stephen Roberts. Blitzkriging: Kronecker-structured stochastic gaussian processes. *arXiv preprint arXiv:1510.07965*, 2015.

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

Jonathan Ozik, Nicholson Collier, Randy Heiland, Gary An, and Paul Macklin. Learning-accelerated discovery of immune-tumour interactions. *Molecular systems design & engineering*, 4(4):747–760, 2019.

Chiwoo Park and Daniel Apley. Patchwork kriging for large-scale gaussian process regression. *The Journal of Machine Learning Research*, 19(1):269–311, 2018.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Benjamin Peherstorfer, Tiangang Cui, Youssef Marzouk, and Karen Willcox. Multifidelity importance sampling. *Computer Methods in Applied Mechanics and Engineering*, 300: 490–509, 2016.

Benjamin Peherstorfer, Boris Kramer, and Karen Willcox. Multifidelity preconditioning of the cross-entropy method for rare event simulation and failure probability estimation. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):737–761, 2018a.

Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3):550–591, 2018b.

Julien Pelamatti, Loic Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Yannick Guerin. Bayesian optimization of variable-size design space problems. *Optimization and Engineering*, 22(1):387–447, 2021.

Victor Picheny and David Ginsbourger. Noisy kriging-based optimization methods: a unified implementation within the diceoptim package. *Computational Statistics & Data Analysis*, 71:1035–1053, 2014.

Victor Picheny, David Ginsbourger, Olivier Roustant, Raphael T Haftka, and Nam-Ho Kim. Adaptive designs of experiments for accurate approximation of a target region. *Journal of Mechanical Design*, 132(7), 2010.

Victor Picheny, Robert B Gramacy, Stefan Wild, and Sébastien Le Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented lagrangian. In *Advances in neural information processing systems*, pages 1435–1443, 2016.

Geoff Pleiss, Jacob Gardner, Kilian Weinberger, and Andrew Gordon Wilson. Constant-time predictive distributions for Gaussian processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4114–4123, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Matthew Plumlee and Rui Tuo. Building accurate emulators for stochastic simulations via quantile kriging. *Technometrics*, 56(4):466–473, 2014.

Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78:1481 – 1497, 10 1990.

Matthew T Pratola, Ofir Harari, Derek Bingham, and Gwenn E Flowers. Design and analysis of experiments on nonconvex regions. *Technometrics*, 59(1):36–47, 2017.

Peter ZG Qian. Sliced latin hypercube designs. *Journal of the American Statistical Association*, 107(497):393–399, 2012.

Peter ZG Qian and CF Jeff Wu. Sliced space-filling designs. *Biometrika*, 96(4):945–956, 2009.

Joaquin Quiñonero and Carl Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.r-project.org/.

Malur R Rajashekhar and Bruce R Ellingwood. A new look at the response surface approach for reliability analysis. *Structural safety*, 12(3):205–220, 1993.

Pritam Ranjan, Derek Bingham, and George Michailidis. Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, 50(4):527–541, 2008.

Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, jan 2006.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

Olivier Roustant, David Ginsbourger, and Yves Deville. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51(1):1–55, 2012. URL https://www.jstatsoft.org/v51/i01/.

Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

Jerome Sacks, Susannah B Schiller, and William J Welch. Designs for computer experiments. *Technometrics*, 31(1):41–47, 1989a.

Jerome Sacks, William Welch, Toby J. Mitchell, and Henry Wynn. Design and analysis of computer experiments. With comments and a rejoinder by the authors. *Statistical Science*, 4, 1989b.

Thomas Santner, Brian Williams, and William Notz. *The Design and Analysis Computer Experiments*. Springer; 2nd edition, 2018.

Manuel Schürch, Dario Azzimonti, Alessio Benavoli, and Marco Zaffalon. Recursive estimation for sparse gaussian process regression. *Automatica*, 120:109127, 2020.

Matthias Seeger, Christopher K I Williams, and Neal D Lawrence. Fast forward selection to speed up sparse Gaussian process regression. *Proceedings - 9th International Conference on Artificial Intelligence and Statistics (AISTATS 2003)*, 9:2003, 2003.

Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Mustererkennung 2000*, pages 27–34. Springer, 2000.

Burr Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 1–18, 2011.

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

Alex J Smola and Peter L Bartlett. Sparse greedy Gaussian process regression. In T K Leen, T G Dietterich, and V Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

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.

Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank gaussian process regression. *Statistics and Computing*, 30(2):419–446, 2020.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

Rajan Srinivasan. *Importance sampling: Applications in communications and detection*. Springer Science & Business Media, 2013.

Michael L Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

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.

Furong Sun, Robert B Gramacy, Benjamin Haaland, Earl Lawrence, and Andrew Walker. Emulating satellite drag from large simulation experiments. *IAM/ASA Journal on Uncertainty Quantification*, 7(2):720–759, 2019a. preprint arXiv:1712.00182.

Furong Sun, Robert B Gramacy, Benjamin Haaland, Siyuan Lu, and Youngdeok Hwang. Synthesizing simulation and field data of solar irradiance. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 12(4):311–324, 2019b.

Chih-Li Sung, Robert B Gramacy, and Benjamin Haaland. Exploiting variance reduction potential in local Gaussian process search. *Statistica Sinica*, 28:577–600, 2018.

Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets–optimization test problems (sum of different powers function). sfu. ca/~ ssurjano/index. html. *Or just google sum of different powers function*, 2014.

Daniel Heestermans Svendsen, Luca Martino, and Gustau Camps-Valls. Active emulation of computer codes with gaussian processes–application to remote sensing. *Pattern Recognition*, 100:107103, 2020.

Linda SL Tan, Victor MH Ong, David J Nott, and Ajay Jasra. Variational inference for sparse spectrum gaussian process regression. *Statistics and Computing*, 26(6):1243–1261, 2016.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. PMLR.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Alexander Usynin, J Wesley Hines, and Aleksey Urmanov. Uncertain failure thresholds in cumulative damage models. In *2008 Annual Reliability and Maintainability Symposium*, pages 334–340. IEEE, 2008.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, New York, NY, 2013.

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

Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: an o (n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, volume 1, pages 288–293, 2000.

Grace Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia, 1990.

Hongqiao Wang and Jinglai Li. Adaptive gaussian process approximation for bayesian inference with expensive likelihood functions. *Neural computation*, 30(11):3072–3094, 2018.

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, pages 14622–14632, 2019.

Peng Wang and David W Coit. Reliability and degradation modeling with random or uncertain failure threshold. In *2007 Annual Reliability and Maintainability Symposium*, pages 392–397. IEEE, 2007.

James Warner, Patrick Leser, William Leser, D Austin Cole, and Geoffrey Bomarito. Assessing next-gen spacesuit reliabilty: a Ppobabilistic analysis case study. Technical report, 2021.

Christopher J Werner, Jeffrey S Bull, CJ Solomon, Forrest B Brown, Gregg W McKinney, Michael E Rising, David A Dixon, Roger L Martz, Henry G Hughes, Lawrence J Cox, et al. Mcnp6. 2 release notes: Report la-ur-18-20808. *Los Alamos National Laboratory*, 2018.

Christopher K I Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In T K Leen, T G Dietterich, and V Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

Andrew Gordon Wilson and Hannes Nickisch. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *Proceedings of the 32Nd International Conference on*

International Conference on Machine Learning - Volume 37, ICML'15, pages 1775–1784. JMLR.org, 2015.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Brian A Worley. Deterministic uncertainty analysis. Technical report, 12 1987.

Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. In *Advances in Neural Information Processing Systems*, pages 5267–5278, 2017.

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

Guangrui Xie and Xi Chen. A heteroscedastic t-process simulation metamodeling approach and its application in inventory control and optimization. In *2017 Winter Simulation Conference (WSC)*, pages 3242–3253. IEEE, 2017.

Boya Zhang, D Austin Cole, and Robert B Gramacy. Distance-distributed design for gaussian process surrogates. *Technometrics*, pages 1–13, 2019.

Boya Zhang, Robert B Gramacy, Leah Johnson, Kenneth A Rose, and Eric Smith. Batch-sequential design and heteroskedastic surrogate modeling for delta smelt conservation. *arXiv preprint arXiv:2010.06515*, 2020a.

Yichi Zhang, Daniel W Apley, and Wei Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports*, 10(1):1–13, 2020b.

Shun-Peng Zhu, Qiang Liu, and Hong-Zhong Huang. Probabilistic modeling of damage accumulation for fatigue reliability analysis. *Procedia Structural Integrity*, 4:3–10, 2017.

Appendix

Here we provide more details for re-expressing $\hat{\tau}^{2(\bar{n},m)}$ in equation (4.8) as equation (4.13). Based on Binois et al. (2018a), we can re-write $\hat{\tau}^{2(\bar{n},m)}$ as

$$\hat{\tau}^{2(\bar{n},m)} = N^{-1}\left(\mathbf{Y}_n^\top \Omega_n^{-1(m)} \mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^\top \Lambda_{\bar{n}}^{(m)} \bar{\mathbf{Y}}_{\bar{n}} + \bar{n}\grave{\tau}^{2(\bar{n},m)}\right)$$

where $\grave{\tau}^{2(\bar{n},m)} = \bar{n}^{-1}\bar{\mathbf{Y}}_{\bar{n}}^\top \grave{\mathbf{\Sigma}}_{\bar{n}}^{-1}\bar{\mathbf{Y}}_{\bar{n}}$ and $\grave{\mathbf{\Sigma}}_{\bar{n}}^{(m)} = \mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top + A_{\bar{n}}^{-1}\Omega_{\bar{n}}^{(m)}$. Recall that $\Omega_{\bar{n}}^{(m)} = \Delta_{\bar{n}}^{(m)} + g\mathbb{I}_{\bar{n}}$, containing the diagonal correction term $\Delta_{\bar{n}}^{(m)} = \mathrm{Diag}\{\mathbf{K}_{\bar{n}} - \mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top\}$. $\grave{\mathbf{\Sigma}}_{\bar{n}}^{(m)}$ contains $\Delta_{\bar{n}}^{(m)}$ in the reweighting, treating it as part of the pure noise. A correct expression of $\hat{\tau}^{2(\bar{n},m)}$ with unique-$\bar{n}$ calculations and $\bar{\mathbf{Y}}_{\bar{n}}$ is

$$\bar{\tau}^{2(\bar{n},m)} = \bar{n}^{-1}\bar{\mathbf{Y}}_{\bar{n}}^\top \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1}\bar{\mathbf{Y}}_{\bar{n}},$$

with

$$\bar{\mathbf{\Sigma}}_{\bar{n}}^{(m)} = \mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top + \Delta_{\bar{n}}^{(m)} + gA_{\bar{n}}^{-1}.$$

Now we seek to write $\grave{\tau}^{2(\bar{n},m)}$ in terms of $\bar{\tau}^{2(\bar{n},m)}$ by redefining $\grave{\mathbf{\Sigma}}_{\bar{n}}^{-1}$ in terms of $\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1}$. By using the Woodbury identity for $(\mathbf{B}+\mathbf{CDE})^{-1}$ from (4.5), we let $\mathbf{B} = \bar{\mathbf{\Sigma}}_{\bar{n}}^{(m)} = \mathbf{k}_{\bar{n}m}\mathbf{K}_m^{-1}\mathbf{k}_{\bar{n}m}^\top + \Delta_{\bar{n}}^{(m)} + gA_{\bar{n}}^{-1}$, $\mathbf{D} = (A_{\bar{n}}^{-1} - \mathbb{I}_{\bar{n}})\Delta_{\bar{n}}^{(m)}$, and $\mathbf{C} = \mathbf{E} = \mathbb{I}_{\bar{n}}$. Then

$$\begin{aligned}
\grave{\mathbf{\Sigma}}_{\bar{n}}^{-1} &= (\mathbf{B} + \mathbf{D}) \\
&= \mathbf{B}^{-1} - \mathbf{B}^{-1}(\mathbf{D}^{-1} + \mathbf{B}^{-1})^{-1}\mathbf{B}^{-1} \\
&= \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)} - \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}((A_{\bar{n}}^{-1} - \mathbb{I}_{\bar{n}})^{-1}\Delta_{\bar{n}}^{-1(m)} + \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)})^{-1}\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}
\end{aligned}$$

Therefore, it follows that

$$
\begin{aligned}
\grave{\tau}^{2(\bar{n},m)} &= \bar{n}^{-1}\bar{\mathbf{Y}}_{\bar{n}}^{\top}\grave{\bar{\mathbf{\Sigma}}}_{\bar{n}}^{-1}\bar{\mathbf{Y}}_{\bar{n}} \\
&= \bar{n}^{-1}\bar{\mathbf{Y}}_{\bar{n}}^{\top}\left(\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)} - \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}(\mathbf{D}^{-1} + \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)})^{-1}\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}\right)\bar{\mathbf{Y}}_{\bar{n}} \\
&= \bar{n}^{-1}\left(\bar{n}\bar{\tau}^{2(\bar{n},m)} - \bar{\mathbf{Y}}_{\bar{n}}^{\top}\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}((A_{\bar{n}}^{-1} - \mathbb{I}_{\bar{n}})^{-1}\Delta_{\bar{n}}^{-1(m)} + \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)})^{-1}\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}\bar{\mathbf{Y}}_{\bar{n}}\right).
\end{aligned}
$$

In turn,

$$
\begin{aligned}
\hat{\tau}^{2(\bar{n},m)} &= N^{-1}\left(\mathbf{Y}_n^{\top}\Omega_n^{-1(m)}\mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^{\top}\Lambda_{\bar{n}}^{(m)}\bar{\mathbf{Y}}_{\bar{n}} + \bar{n}\grave{\tau}^{2(\bar{n},m)}\right) \\
&= N^{-1}\Bigg(\bar{n}\bar{\tau}^{2(\bar{n},m)} + \mathbf{Y}_n^{\top}\Omega_n^{-1(m)}\mathbf{Y}_n - \bar{\mathbf{Y}}_{\bar{n}}^{\top}\Lambda_{\bar{n}}^{(m)}\bar{\mathbf{Y}}_{\bar{n}} - \\
&\qquad \bar{\mathbf{Y}}_{\bar{n}}^{\top}\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}((A_{\bar{n}}^{-1} - \mathbb{I}_{\bar{n}})^{-1}\Delta_{\bar{n}}^{-1(m)} + \bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)})^{-1}\bar{\mathbf{\Sigma}}_{\bar{n}}^{-1(m)}\bar{\mathbf{Y}}_{\bar{n}}\Bigg).
\end{aligned}
$$