Scalable Bayesian Optimization Using Vecchia Approximations of Gaussian Processes

Felix Jimenez
Texas A&M University

Matthias Katzfuss
Texas A&M University

Abstract

Bayesian optimization is a technique for optimizing black-box target functions. At the core of Bayesian optimization is a surrogate model that predicts the output of the target function at previously unseen inputs to facilitate the selection of promising input values. Gaussian processes (GPs) are commonly used as surrogate models but are known to scale poorly with the number of observations. Inducing point GP approximations can mitigate scaling issues, but may provide overly smooth estimates of the target function. In this work we adapt the Vecchia approximation, a popular GP approximation from spatial statistics, to enable scalable high-dimensional Bayesian optimization. We develop several improvements and extensions to Vecchia, including training warped GPs using mini-batch gradient descent, approximate neighbor search, and variance recalibration. We demonstrate the superior performance of Vecchia in BO using both Thompson sampling and qUCB. On several test functions and on two reinforcement-learning problems, our methods compared favorably to the state of the art, often outperforming inducing point methods and even exact GPs.

1 INTRODUCTION

Bayesian optimization (BO) is a class of techniques for black-box function optimization [Mockus, 1989]. BO has found success in a variety of areas, including reinforcement learning [Calandra et al., 2016], tuning of machine-learning algorithms [Swersky et al., 2013], A/B testing [Letham et al., 2019] and material discovery [Gómez-Bombarelli et al., 2018].

The basic idea of BO is to approximate the black-box function with a cheap-to-evaluate surrogate, typically a GP, and perform optimization with the surrogate. Starting with an initial collection of function evaluations, the trained GP is combined with a heuristic to generate an *acquisition function*, which assigns values to unseen inputs. The input with the highest acquisition value is evaluated and the result is added to the dataset. This process continues iteratively until a predefined stopping criterion is met or the computational budget is depleted.

As most BO techniques use a GP surrogate, they inherit the scalability issues of GPs. It becomes expensive to search a high-dimensional space when optimizing the acquisition function. This work addresses this scalability issue by extending the Vecchia GP approximation from spatial statistics [Vecchia, 1988, Katzfuss and Guinness, 2021] to work in high dimensions with many observations. Through these extensions, we demonstrate how Vecchia GPs can be used in concert with existing BO procedures. To be precise, our contributions are : (1) Introduce Vecchia GP approximations to the machine-learning community (2) Adapt Vecchia GPs to BO and improve their flexibility and scaling with warped kernels, vector quantization, approximate nearest neighbors, minibatch training, and variance correction (3) Empirically demonstrate the utility of Vecchia GPs within existing BO frameworks.

2 EXISTING SOLUTIONS

There has been interest in scaling BO in two ways: higher input dimensions and more function evaluations. While we focus on the latter here, these two aspects are related, in that successful high-dimensional optimization typically requires many function evaluations.

High-dimensional BO is thoroughly studied with many proposed solutions, including transformations to a lower-dimensional space [Gómez-Bombarelli et al., 2018], assuming an additive form for the target function [Kandasamy et al., 2015], and using sparsity-inducing priors [Eriksson and Jankowiak, 2021].

Scaling BO to many function evaluations is less ex-

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

plored. Existing approaches rely on additivity assumptions [Wang et al., 2018] or GP approximations [McIntire et al., 2016, Maddox et al., 2021]. The weighted online Gaussian process (WGOP) [McIntire et al., 2016] method uses a second sparse GP to learn hyperparameters and as such is not a self-contained method. The online variational conditioning (OVC) [Maddox et al., 2021] method improves the efficiency of conditioning sparse GPs on new data for look-ahead acquisition functions such as batch knowledge gradient (qKG) [Wu and Frazier, 2016], but it does not change the predictive performance of sparse GPs. The trust region BO method (TuRBO) [Eriksson et al., 2019] has also seen great success by relying on Thompson sampling and limiting the input search to trust regions, which are determined by the lengthscale parameters of the GP. However, TuRBO relies on fitting exact GPs to subsets of data within trust regions when using multiple trust regions, or using a global exact GP when there is a single trust region. This means the method is still bound by the computational complexity of exact GPs or it cannot leverage all the available data when making predictions within a trust region.

In this work we propose using Vecchia GPs as a drop-in replacement for exact and sparse GPs within BO. We show that existing GP-based BO methods can be made scalable via the Vecchia approximation, and we establish that existing scalable BO methods can be improved by using Vecchia GPs instead of inducing point methods [Hensman et al., 2013], which can suffer from important limitations [Stein, 2014]. While local information and nearest neighbor conditioning sets have been used for variational approximations to GPs [Liu and Liu, 2019, Tran et al., 2021, Wu et al., 2022], we believe our approach's ease of use and strong performance justify further investigation within the BO community. We hope to provide readers with a set of techniques for easily incorporating Vecchia GPs into existing BO procedures to improve their scalability.

3 BACKGROUND

3.1 Review of Bayesian Optimization

For a "black-box" objective function, f, Bayesian optimization aims to solve the problem

$$\min_{\boldsymbol{x} \in \mathcal{X}} f(x), \tag{1}$$

over some input space $\mathcal{X} \subset \mathbb{R}^d$. In BO, f is often modeled as a Gaussian process (GP), $f(\cdot) \sim GP(\mu_{\boldsymbol{\theta}}, K_{\boldsymbol{\theta}})$, whose mean $\mu_{\boldsymbol{\theta}}$ and kernel function $K_{\boldsymbol{\theta}}$ depend on hyperparameters $\boldsymbol{\theta}$.

Function evaluations $y_i = f(\boldsymbol{x}_i)$ are obtained sequentially. Given the first n observations $\boldsymbol{y}_{1:n}$ at inputs $\boldsymbol{x}_{1:n}$, the goal is to find a new input $\boldsymbol{x}_{n+1}$, such that the global optimum of f is likely at $\boldsymbol{x}_{n+1}$. This is usually achieved by maximizing an acquisition function, $\boldsymbol{x}_{n+1} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} a_{\boldsymbol{\theta}^*}(\boldsymbol{x}|\boldsymbol{y}_{1:n})$, which is a function that assigns a measure of value to unseen locations, and where $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log p_\theta(\boldsymbol{y}_{1:n})$ is the maximum likelihood estimate.

Alternatively, new inputs can be chosen using Thompson sampling [Thompson, 1933], which has been shown to work well empirically [Chapelle and Li, 2011]. In Thompson sampling, we consider a collection of M inputs, $\boldsymbol{X} = (\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_M^*)$, sampled from the input space $\mathcal{X}$. We generate a joint sample from the posterior at these locations $(y_1^*, \ldots, y_M^*) \sim p_{\boldsymbol{\theta}}(f(\boldsymbol{X})|\boldsymbol{y}_{1:n})$. We choose $\boldsymbol{x}_{n+1}$ to be the input value with the lowest sampled value, $\boldsymbol{x}_{n+1} = \boldsymbol{x}_j^\star$ with $j = \arg\min_{i \in \{1,\ldots,M\}} y_i^*$. If we want a batch of $q > 1$ query points, we generate q independent batches from the posterior over the same set $\boldsymbol{X}$ and choose the lowest value from each batch (avoiding duplicates).

3.2 Vecchia GP Approximations

Given n observations $\boldsymbol{y}_{1:n}$ as described above, the Vecchia GP approximation [Vecchia, 1988, Katzfuss and Guinness, 2021] uses a modified likelihood to turn the $\mathcal{O}(n^3)$ complexity for standard GP regression into $\mathcal{O}(nm^3)$, with $m \ll n$. (A further reduction to $\mathcal{O}(nm^2)$ is possible by grouping observations and re-using Cholesky factors [Schäfer et al., 2021].) The following provides details on Vecchia GPs, including the likelihood and the posterior predictive distribution.

3.2.1 Likelihood

Recall that any joint density of n observations $\boldsymbol{y}_{1:n}$ can be decomposed exactly as a product of conditional densities: $p(\boldsymbol{y}_{1:n}) = \prod_{i=1}^n p(y_i|\boldsymbol{y}_{1:i-1})$, where $\boldsymbol{y}_{1:0} = \emptyset$. This motivates the Vecchia approximation [Vecchia, 1988]:

$$p(\boldsymbol{y}_{1:n}) \approx \widehat{p}(\boldsymbol{y}_{1:n}) = \textstyle\prod_{i=1}^n p(y_i|\boldsymbol{y}_{c(i)}), \tag{2}$$

where each $c(i) \subset \{1, \ldots, i-1\}$ is a conditioning index set of size at most m, and the n conditional distributions $p(y_i|\boldsymbol{y}_{c(i)})$ are all Gaussian and can be computed in parallel, each in $\mathcal{O}(m^3)$ time.

3.2.2 Properties

Vecchia approximations have several attractive properties [Katzfuss et al., 2022]. The joint distribution $\widehat{p}(\boldsymbol{y}_{1:n}) = \mathcal{N}(\boldsymbol{\mu}, \widehat{\boldsymbol{K}})$ implied by (2) is multivariate Gaussian, where the inverse Cholesky factor $\widehat{\boldsymbol{K}}^{-1/2}$ is sparse with $\mathcal{O}(nm)$ nonzero entries [Katzfuss and Guinness, 2021]. Under the sparsity pattern implied by the $c(i)$, the Vecchia approximation results in the optimal $\widehat{\boldsymbol{K}}^{-1/2}$ as measured by Kullback-Leibler (KL) divergence, $KL(p(\boldsymbol{y})\|\widehat{p}(\boldsymbol{y}))$ [Schäfer et al., 2021].

In general the matrix $\widehat{\boldsymbol{K}}$ is dense but for GP hyperparameter optimization and prediction we never need to explicitly compute $\widehat{\boldsymbol{K}}$. Instead we work with another sparse matrix $\boldsymbol{U}$. $\boldsymbol{U}$ is the the upper-lower Cholesky decomposition of $\widehat{\boldsymbol{K}}$, which is simply the inverse Cholesky factor of $\boldsymbol{P}_c\boldsymbol{P}_r\widehat{\boldsymbol{K}}$. Where $\boldsymbol{P}_c$ and $\boldsymbol{P}_c$ are permutation matrices that reverse columns and rows, respectively. The elements of $\boldsymbol{U}$ can be computed in parallel and involve inverting $m \times m$ matrices (see Appendix D.1 for closed form expressions).

Growing the conditioning sets $c(i)$ decreases the KL divergence [Guinness, 2018]; for $m = n - 1$, the approximation becomes exact, $\widehat{p}(\boldsymbol{y}) = p(\boldsymbol{y})$. Thus, m trades off low computational cost (small m) and high accuracy (large m); crucially, high accuracy can be achieved even for $m \ll n$ in many settings.

3.2.3 Ordering and Conditioning Sets

Aside from the choice of m, the accuracy of a Vecchia approximation depends on the ordering of $\boldsymbol{y}_{1:n}$ and on the choice of the conditioning sets $c(i)$ in (2).

In our experience, the highest accuracy for a given m can be achieved by combining maximum-minimum-distance (maximin) ordering with conditioning on the m (previously ordered) nearest neighbors (NNs), which is illustrated in Figure 1. Maximin sequentially picks each variable in the ordering as the one that maximizes the minimum distance to previously ordered variables. Exact maximin ordering and NN conditioning can be computed in quasilinear time in n, and m needs to grow only polylogarithmically with n for ϵ-accurate Vecchia approximations with maximin ordering and NN conditioning under certain regularity conditions [Schäfer et al., 2021].

In Figure 1, we see that for index i in the maximin ordering, the m previously ordered NNs can be far away for small i and very close for large i. This global and local behavior helps Vecchia both learn accurate values for hyperparameters and make good predictions. This is in contrast to low-rank approximations that effectively use the same conditioning sets for each i and hence focus on global behavior at the cost of ignoring local structure.

The global-local nature of Vecchia can persist even with other variable orderings, such as total random ordering. This is demonstrated in Appendix A.1, where we compare Vecchia (random ordering) and SVI-GP on functions with high frequency fluctuations and sharp minima. On the Michalwicz function [Molga and Smutnicki, 2005], Vecchia outperformed SVI-GP by several orders of magnitude in terms of KL-divergence between the approximate GPs and the exact GP. We attribute much of this success to Vecchia's ability to model local fluctuations, even with random ordering, as we further demonstrate on the Ackley-1 function in Appendix A.1.1.

3.2.4 Vecchia Predictions

The Vecchia approximation naturally allows for an accurate approximation of the joint posterior predictive distribution of an unobserved n_p-vector $\boldsymbol{y}^p$ [Katzfuss et al., 2020] of the form

$$\widehat{p}(\boldsymbol{y}^p|\boldsymbol{y}_{1:n}) = \textstyle\prod_{i=1}^{n_p} p(y_i^p|\boldsymbol{y}^p_{c_p(i)}), \tag{3}$$

where $\boldsymbol{y}^p_{c_p(i)}$ denotes the m NNs of y_i^p among $\boldsymbol{y}_{1:n}$ (or $\boldsymbol{y}_{1:n} \cup \boldsymbol{y}^p_{1:i-1}$, if joint predictions are desired), and the n_p conditional distributions $p(y_i^p|\boldsymbol{y}^p_{c_p(i)})$ are all Gaussian and can be computed in parallel, each in $\mathcal{O}(m^3)$ time.

To be precise, the joint posterior predictive distribution is Gaussian with a sparse inverse Cholesky factor, $\widehat{p}(\boldsymbol{y}^p|\boldsymbol{y}_{1:n}) = \mathcal{N}(\boldsymbol{\mu}_p, (\boldsymbol{L}^T\boldsymbol{L})^{-1})$. Where the mean and inverse Cholesky factor are given by,

$$\boldsymbol{\mu}_p = (\boldsymbol{U}_{p,p}^T)^{-1}(\boldsymbol{U}_{n,p})^T\boldsymbol{y}_{1:n}, \qquad \boldsymbol{L} = \boldsymbol{U}_{p,p}^T,$$

With the entries of $\boldsymbol{U}$ computed as in Equation 11. See Appendix D for additional details on prediction.

3.3 Comparison with Local Expert Models

It should be observed that the Vecchia approximation is different than approximations that rely on local experts [Tresp, 2000, Yuan and Neubauer, 2008, Cao and Fleet, 2014]. We briefly highlight the main differences between the Vecchia approximation and local experts.

With local experts, the data is partitioned into subsets on which local models are trained. At test time the predictions from each model are combined into a single prediction. However, with Vecchia we never partition the data and do not require a combining step when making predictions.

The assumptions on the structure of the covariance and precision matrix are also different between local experts and Vecchia. For example, Product of Expert (PoE) based models assume zeros in the covariance matrix (often a block diagonal structure depending on the ordering of the data), but Vecchia assumes zeros in the inverse of the Cholesky factor. There has been work on allowing correlation between the blocks in PoE models [Schürch et al., 2023], but the focus is still on zeros in the covariance matrix rather than zeros in the inverse Cholesky of the covariance matrix.

4 VECCHIA FOR BAYESIAN OPTIMIZATION

In this section we detail how Vecchia GPs work naturally with standard BO GP modeling techniques. We also work out how Vecchia can be modified to improve performance for BO.

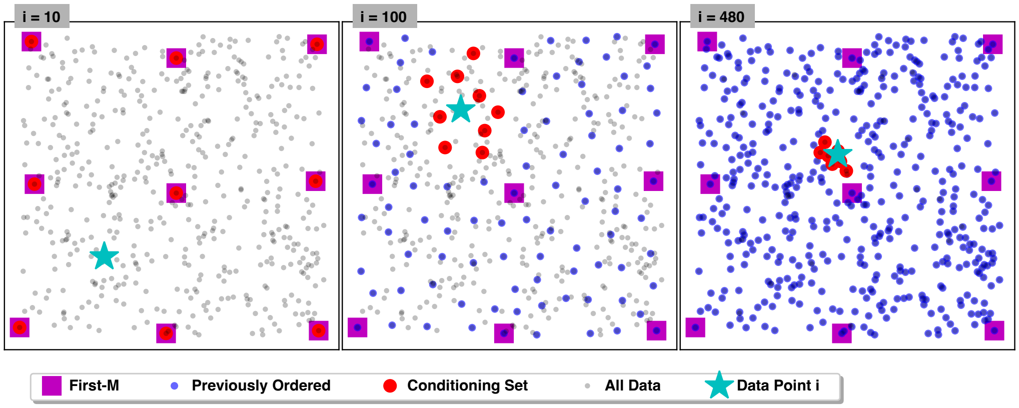

Figure 1: Illustration of maximin ordering and conditioning sets for $n = 500$ uniformly sampled inputs (grey dots) on $[0, 1]^2$. For $i = 10$, $i = 100$, and $i = 480$ in the left, middle, and right panel, we show the ith input (cyan star) in the maximin ordering, the previously ordered $i - 1$ inputs (blue dots), including the (approximate) nearest $m = 9$ inputs (red circles). For increasing i, the first i inputs form increasingly dense grids, so that the distance to the m NN decreases gradually and systematically. In contrast, for fully independent conditional approximations (FIC), each input has the same conditioning set, here illustrated by the first m inputs in the maximin ordering (magenta squares). Thus FIC effectively ignores many closer previously-ordered inputs, which can lead to large approximation errors.

4.1 Warped Kernel

It has been shown that input warping can improve BO performance [Snoek et al., 2014], so we work out how Vecchia can be altered to incorporate warping. We start by explaining the idea of warping and how it can affect Vecchia.

A flexible GP kernel can be obtained by taking a base kernel $\tilde{K}$ (e.g., Matérn 5/2) and warping the inputs with a nonlinear function $w(\cdot)$,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tilde{K}(\|w(\boldsymbol{x}) - w(\boldsymbol{x}')\|). \tag{4}$$

The warping function we focus on is the Kumarswamy CDF [Balandat et al., 2020]. For $\boldsymbol{x} \in \mathbb{R}^d$ we allow each dimension, x_i, to be warped using different parameters, so for the Kumarswamy CDF we have $w(x_i) = 1-(1-x_i^{a_i})^{b_i}$ for $i = 1, ..., d$. When $a_i = b_i = 1$ for all i we simply return $\boldsymbol{x}$. This choice is arbitrary and one could use other warping functions such as a neural network [Wilson et al., 2016].

To fully utilize the flexibility of the warped kernel within Vecchia GPs, we carry out the ordering and conditioning-set selection based on Euclidean distance between the warped inputs $w(\boldsymbol{x}_1), \ldots, w(\boldsymbol{x}_n)$ instead of the original inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

4.2 Training Using Stochastic Gradient Descent

Let $\boldsymbol{\theta}$ denote the vector of hyperparameters for the kernel function $K_{\boldsymbol{\theta}}$ which may include parameters for a warping function $w(\cdot)$. Fitting the Vecchia GP to data involves finding the possibly high dimensional $\boldsymbol{\theta}^*$ such that,

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \Theta}{\arg\max} \sum_{i=1}^{n} \log p(y_i | \boldsymbol{y}_{c(i)})$$

An approximate solution is generally found using gradient descent to optimize the sum of log terms. This approach has a time complexity that is linear in n. However, each term in the sum can be evaluated in $\mathcal{O}(m^3)$ time. We use this observation to approximate the gradient using mini-batches indexed by $b \subset \{1, ..., n\}$:

$$\nabla_{\boldsymbol{\theta}} \log \widehat{p}(\boldsymbol{y}_{1:n}) \approx \textstyle\sum_{j \in b} (n/|b|) \nabla_{\boldsymbol{\theta}} \log p(y_j | \boldsymbol{y}_{c(j)}). \tag{5}$$

With this change, the training time depends on the mini-batch size b instead of the total data size n given the ordering and NN. This reduction in time complexity is critical, as we are estimating the GP hyperparameters after each step in the BO procedure.

Empirically we have observed notable speed up in training time and consistent results regardless of batch size. Appendix G explores the results of different batch sizes when fitting data from the Ackley-25 function with 1500 observations. The final loss achieved was consistent for all batch sizes, but for the smallest batch size, namely 32, the result was achieved in many fewer passes through the data.

4.3 Nearest Neighbors and Ordering

The nearest-neighbor search can become a bottleneck in the Vecchia procedure, and to alleviate this we suggest using approximate NNs with product quantization [Jegou et al., 2010, Johnson et al., 2017]. At a high level constructing the conditioning sets proceeds as follows, given a database to query against we begin by constructing a set of centroids of length N_{list}. Then we create Voronoi cells around each of these centroids and cluster the database points based on which Voronoi cell they fall into. To find the approximate conditioning set $\tilde{c}(i^*)$ for test point $\boldsymbol{x}_i^*$ we compare the query point $\boldsymbol{x}_i^*$ against the centroids. We search among data clustered into the N_{probe} closest Voronoi cells and return the m closest data points found. The two parameters N_{list} and N_{probe} allow us to trade off speed and accuracy when constructing our conditioning sets. If N_{list} is large and N_{probe} is small the query will be fast, but we are susceptible to a problem known as the ”edge effect” where a query point is close to the boundary of a Voronoi cell. We explore the impact of N_{list} and N_{probe} on the performance of Vecchia GPs in Section 5.4.

For variable ordering a batched maximin ordering (see Appendix B) is natural for BO since we observed the data in batches. By preserving the order of the previously seen data points the complexity of ordering can be reduce to quasilinear in the query size.

4.4 Calibration of Predictive Variances

To improve uncertainty quantification for out-of-sample predictions, we calibrate the marginal predictive variances [Katzfuss et al., 2022]. In short, we maximize the log-probability of the most recent query set by adding a term to the diagonal of the covariance matrix.

To do this we update the hyperparameters based on all available data, and then compute the posterior for the newly observed query points as described in Section 3.2.4. Finally, we choose the variance inflation factor, b_v, to maximize the posterior log-probability of this holdout set:

$$\max_{b_v \in [0,2]} \log p(\tilde{\boldsymbol{y}} | \tilde{\boldsymbol{\mu}}_{\boldsymbol{\theta}^*}, \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\theta}^*} + b_v \boldsymbol{I}_{\tilde{n}}). \tag{6}$$

More details can be found in Algorithm 2 in Appendix C.

With $b_v > 0$, the predictive variance is inflated, which encourages more exploration in the resulting BO procedure. Since the observations are processed to have zero mean and variance equal to one, the value of b_v is generally at most 1.4 but usually less than 0.1. In our experience, this variance correction is helpful when the observations are noisy or the target function is highly variable. However, if the function is smooth and we have noise-free observations, simply using the initial Vecchia predictions is sufficient. In our experiments we found the optimization of the log-probability faster with additive variance inflation, as opposed to multiplicative variance inflation which has been used in previous work [Katzfuss et al., 2022].

4.5 Speeding Up Existing BO Methods

Below we provide details for incorporating Vecchia GPs into existing BO frameworks to improve their computational speed.

4.5.1 Vecchia with TuRBO

Trust Region Bayesian optimization (TuRBO) [Eriksson et al., 2019] is a Bayesian optimization procedure that uses hyper-rectangles called *trust regions*. In TuRBO, independent exact GPs are fit to data within the trust regions. Query points are chosen using Thompson sampling, and the regions grow or shrink based on how often they contain new optima. TuRBO allows for a single trust region (TuRBO-1) or $t > 1$ simultaneous trust regions (TuRBO-t). TuRBO-1 is based on a single global GP fit to all data, whereas TuRBO-t fits t local GPs, each using only the data within one of the trust regions.

Veccchia fits in the TuRBO framework without much change. Using either TuRBO-1 or TuRBO-t, the same (global) Vecchia GP is shared between all trust regions, and predictions within each trust region are made using the m NNs among all data (i.e., the NNs are not restricted to be in the trust region). By conditioning on the NNs, we emphasize the local behavior of the target function, and by training on all the data we ensure our hyperparameters are well calibrated. For the remainder of this paper, we are referring to TuRBO-1 when we say TuRBO.

4.5.2 Vecchia for Optimizing Parallel Acquisition Functions with Gradients

To perform the inner optimization step when using parallel acquisition functions, such as q-EI or q-UCB, we need to estimate the gradient of the acquisition function, $\nabla \mathcal{L}(\mathbf{X})$. This gradient can be estimate using Monte Carlo [Wilson et al., 2018]:

$$\nabla \mathcal{L}(\mathbf{X}) \approx \frac{1}{m} \sum_{k=1}^{m} \nabla \ell(\mathbf{y}^k), \tag{7}$$

where $\ell(\mathbf{y}^k)$ is the acqusition function’s corresponding utlity function, and the $\mathbf{y}^k$’s are draws from the posterior over f. For several parallel acquisition functions, the convergence of the corresponding stochastic optimization requires, among other things, that the mean and covariance functions of the GP be continuously differentiable for each point in the domain [Wang et al., 2016, Wu and Frazier, 2016]. This is guaranteed to be satisfied for Vecchia GPs except for a set of points of measure zero, and optimizing the acquisition function using standard gradient descent works well in practice. For a detailed discussion on the impact of these discontinuities in Vecchia predictions see Appendix M.2.

Vecchia can improve the speed of the Monte Carlo estimate, by allowing for fast posterior sampling of the $\mathbf{y}^k$'s (see Section 3.2.4). In this work we use qUCB [Wilson et al., 2017] as our acqusition function.

5 NUMERICAL COMPARISONS

We evaluated Vecchia GPs with qUCB and TuRBO on a set of benchmarks [Balandat et al., 2020]. The synthetic functions were Ackley-5, Hartmann-6, Branin-2 and Levy. For the Levy function we used what we referred to as the Levy 55-20 function, where we warp the space with a sigmoid type function and add irrelevent dimensions. Finally, we used two reinforcement-learning examples, namely robot pushing [Wang et al., 2018] and a lunar lander problem in OpenAI's gym environment. For the Lunar Lander we must tune a PID controller [Eriksson et al., 2019] to successfully land a Lunar module at a fixed point on a set of 50 maps. In the 14-dimensional robot-pushing problem [Wang et al., 2018], there are two robot arms pushing objects towards a target location. We must minimize the distance to the target location. Experiment using the Rastrigin function in 20 and 100 dimensions using budgets of 1,000 and 35,000 are given in Appendix K.

5.1 GP Surrogates

We compared the following GP surrogates:

Exact GP: A standard Gaussian process, which is very slow for large n.

LR-First m: Low-rank approximation where conditioning set was taken as the first m points in the maximin ordering, the same for all n data points.

SVI-GP: The SVI GP [Hensman et al., 2013] places a Gaussian variational distribution on the outputs of "pseudo-points". The variational parameters, including the location of the m inducing inputs, are learned with stochastic mini-batch natural gradient descent.

WOGP: The weighted online Gaussian process [McIntire et al., 2016] is an online inducing point method that attempts to minimize a weighted KL-divergence between a GP that incorporates all the new data and a reduced GP. The weighted KL-divergence encourages the reduced GP to match the larger GP near input of the current maximum in the training set. The m inducing input locations are chosen from the data during this online learning procedure. The method relies on an auxiliary GP to first estimate the GP hyperparameters and then uses these fixed hyperparameters when performing online learning. For WOGP hyperparameters we used the hyperparameters from the SVI-GP trained to the intial space filling observations. Our code is based on the original authors' implementation: `https://github.com/ermongroup/bayes-opt`.

Vecchia: Our method as described in the previous sections, where the likelihood is approximated by conditioning each observation on the m NN subject to the constraint that the neighbors appear before the data point in the maximin ordering. Code for Vecchia GPs is available at `https://github.com/feji3769/VecchiaBO`.

For the approximate methods, we let m increase polylogarithmically with n, as $m = C_0 * \log_{10}^2(n)$. The value of C_0 was set to 7.2; for details about this selection see Appendix F.2. Each model uses a Matérn 5/2 kernel as the base kernel $\tilde{K}$ in (4). For all experiments except the Levy example, we used an affine warping, resulting in an automatic relevance determination kernel K. For the Levy example, in which the input space was purposely warped, the warping function was taken to be the Kumarswamy CDF.

5.2 Comparison Measures

For target functions whose optimum is not known, we simply used the best value found as our measure of performance. For functions whose optimum is known, we used *regret* as our measure of performance. Regret after t function evaluations is given by,

$$r_t = \min_{1 \leq i \leq t} \{y_i - f^*\}, \tag{8}$$

where f^* is the true minimum and the min is taken over all previously observed function values. Our regret curves are averaged over independent repeats, and the bands around each curve represent a 95% C.I. for the mean.

5.3 Results

5.3.1 Hartmann-6 and Ackley-5

The left panel of Figure 2 shows the Vecchia GP doing the best among all approximate GPs on optimizing the Hartmann-6 function. In the right panel of Figure 2 we see Vecchia matching the performance of the exact GP on the Ackley-5 function, and again beating all other approximate GPs.

The strong performance of Vecchia on Ackley-5 is in line with our analysis in Appendix A.1.1, where Vecchia outperformed SVI-GP on the Ackley-1 function. In that experiment we observed that Vecchia was able to capture the high frequency fluctuations of the target function better than the inducing point based SVI-GP. The local nature of Vecchia naturally complements the TuRBO algorithm, and BO in general.

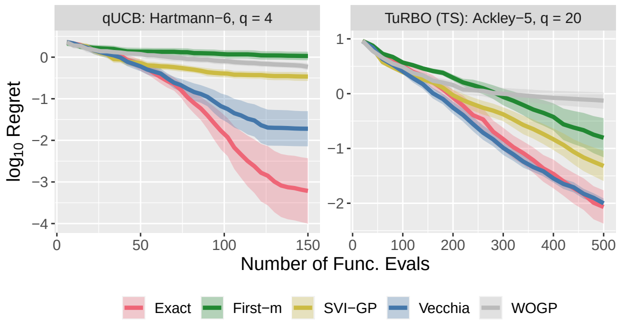

Figure 2: Mean and 95% C.I. of log regret for combination of GP surrogates and acquisition functions. All GPs use the base kernel, $\tilde{K}$. **Left**: Hartmann-6 using qUCB with $q = 4$, Vecchia does the best among all approximate GPs. **Right**: Ackley-5 using TuRBO with batch size $q = 20$, Vecchia matches the exact and again outperforms all approximate GPs.

5.3.2 Branin and Levy

For the Branin-2 function the left panel of Figure 3 shows that including variance inflation for Vecchia improved the results. Both inflated and non-inflated Vecchia still outperform all other approximate GP surrogates. In the right panel of Figure 3 we see that Vechia without warping outperforms all GP surrogates, including the exact, and by including warping Vecchia performs even better. While warping for the exact GP actually hurt results. We believe that the warping did not improve the exact GP the same way it improved Vecchia because Vecchia with warping is making the best local approximations of the function possible. Due to the NN conditioning, Vecchia was able to focus on local behavior of this relatively complex function and ignored irrelevant distant locations.

5.3.3 Lunar Lander and Robot Pushing

For the 14D Robot Pushing problem Figure 4 shows Vecchia performed the best among all methods, eventually even outperforming the exact GP for a large number of function evaluations. In this experiment Vecchia GP included variance inflation, and the variance inflation pushed Vecchia just beyond the exact GP. As for the 12D Lunar Lander, the left panel in Figure 4 shows that Vecchia and SVI-GP performed best. Both outperformed the exact GP, with SVI-GP just passing Vecchia. Just as with the 14D robot pushing problem, the variance inflation helped Vecchia's final performance.

We believe noisy functions and those with many local optima are the most likely to benefit from variance inflation. Our reasoning is that variance inflated Vecchia, much like an epsilon greedy policy, injects a small amount of randomness into the search procedure. This encourages evaluating points that may not have been likely under the non-inflated Vecchia posterior. However, since the inflation factor is based on predictive performance, as more data is collected we inflate less and therefore explore less.

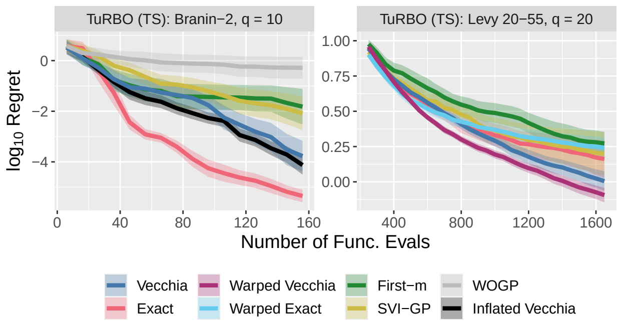

Figure 3: Mean and 95% C.I. of log regret for combination of GP surrogates, kernels and variance correction. **Left**: Branin-2 using TuRBO $q = 10$, all GPs use the base kernel, $\tilde{K}$. Vecchia outperforms all the approximate surrogates. **Right**: Warped and embedded 20-55 Levy using TuRBO with batch size $q = 20$. The warped variants of Vecchia and exact, use the warped kernel in Equation 4 with the Kumarswamy CDF, while all other GPs use the base kernel, $\tilde{K}$. Vecchia and Warped Vecchia outperform all other surrogates.

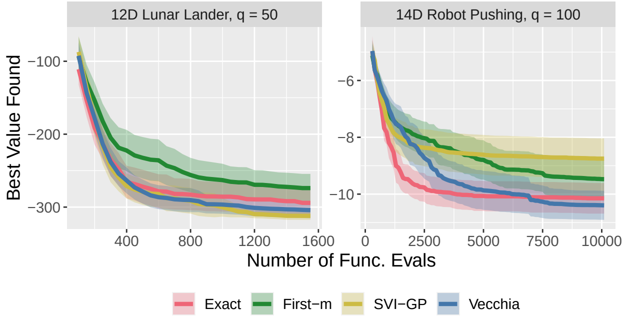

Figure 4: Mean and 95% C.I. of the best value found for combination of GP surrogates with the same base kernel, $\tilde{K}$. **Left** 12D Lunar Lander problem with batch size $q = 50$. Both Vechia and SVI-GP outperform the exact. **Right** 14D Robot Pushing problem with batch size $q = 100$. Vecchia bests all approximate GP methods and surpasses the exact GP by 7500 evaluations.

5.4 Ablation Study

We now investigate how warping and the approximate nearest neighbors, introduced in Section 4, affect Vecchia's performance in estimating the function of interest. To quantify the performance of Vecchia predictions we use either mean squared error (MSE) or negative log likelihood (NLL).

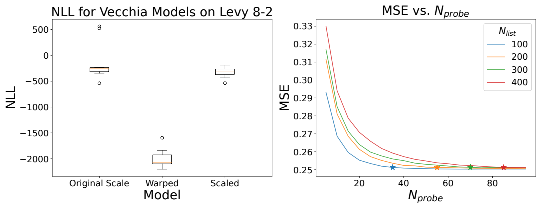

Figure 5: **Left** Box plots showing the NLL for Vecchia GPs (lower is better) using the original input space, a warped space and the scaled space for Ackley-10. Warping results in the best performance among the three. **Right** The MSE of Vecchia GPs (lower is better) with approximate nearest neighbors vs N_{probe}. The colors denote different values of N_{list} and the stars represent the average value of N_{probe} such that 70% of the m nearest neighbors are correctly identified for a given N_{list}.

The left plot in Figure 5 shows the NLL for three different Vecchia GPs: warped Vecchia, scaled Vecchia and standard response first Vecchia. In this experiment we generate data from a problem that is purposefully sparse using the Levy-2 function. For more details see Appendix L.1. The plots suggest that when the true function is low dimensional, warping can improve results over other input transforms. It may be that the strong performance of warped Vecchia on the Levy-55-20 function can be attributed to the low dimensional nature of the inputs.

The right plot in Figure 5 shows the effect of the parameters N_{list} and N_{probe} when using approximate nearest neighbors to construct conditioning sets. For this experiment we use 300,000 observations of the Ackley-20 function for training and we vary N_{list} and N_{probe} to observe how these parameters affect the MSE of the Vecchia predictions. For more details on this experiment see Appendix L.2. It's worth pointing out two features of this plot. The first is that we can get similar performance as exact nearest neighbors using fast approximations. The second thing is that we can use the recall (how many of the exact nearest neighbors are recovered) of the approximate nearest neighbors as a way to guide our choice of N_{list} and N_{probe}. That is as long as a high proportion of the m nearest neighbors are recovered correctly, then we should expect the approximate-NN conditioning sets to provide similar performance to the exact-NN conditioning sets.

6 DISCUSSION

We explored the use of approximate Vecchia GPs for speeding up Bayesian optimization (BO). We described how training of Vecchia GPs can be made efficient through mini-batch gradient descent, approximate nearest neighbors and approximate orderings. We examined how these changes can affect the performance of Vecchia GPs on sparse and large scale problems. Additionally, we incorporated Vecchia GPs within existing BO procedures.

In our numerical experiments we found Vecchia GPs to often be as or even more accurate than other GP surrogates, including the exact GP. This is remarkable, in that, the Vecchia approximation not only improves regret, but it drastically reduces the computational complexity relative to the exact GP, with a computational cost that largely depends (linearly) on the mini-batch size. As a result, Vecchia can be even computationally cheaper than approximate pseudopoint methods such as SVI-GP, which must optimize over a large number of hyperparameters related to the pseudopoints, especially in high-dimensional input spaces, while Vecchia does not include any additional hyperparameters beyond those already present in the exact-GP model to be approximated. We conjecture that Vecchia's accuracy improvements are at least partially due to the global-local nature of Vecchia GPs; the nearest-neighbor portion of Vecchia focuses on the local behavior of the target function during prediction while using the entire dataset to estimate the hyperparameters from the global information. Together, Vecchia GPs naturally balance local and global information when estimating the target function.

Our results suggest that Vecchia GPs can be useful for speeding up BO when many evaluations of the target function are feasible and necessary, and we are hopeful that Vecchia approximations can be a standard tool for BO in such settings. The primary limitations of Vecchia for BO mainly rely on ordering and nearest neighbor selection, but the ideas presented in Section 3.1 address these issues well enough for many applications.

There are several potential avenues for future work. Vecchia GPs can be used to improve the $\mathcal{O}(dn^3)$ time complexity of sparse axis-aligned subspace BO (SAASBO) [Eriksson and Jankowiak, 2021], which uses horseshoe-like priors on the lengthscale parameters of the GP. In particular, the use of data-splitting [Neal and others, 2011] can be employed to render $\mathcal{O}(dm^3)$ time complexity once nearest neighbors have been computed.

Another possible direction of work is constructing conditioning sets in high dimensional spaces when there is no low dimensional latent structure to exploit. In Appendix L.3 we saw that the scaled Vecchia GP with Euclidean nearest neighbors can readily exploit a low dimensional latent space, but may require a large budget when the latent space is truly high dimensional. While correlation based conditioning sets [Kang and Katzfuss, 2021] are an existing solution to this problem, we believe more work can be done to improve Vecchia predictions for BO tasks with high dimensional latent spaces.

For highly noisy target functions, we believe that further accuracy gains can be made by latent Vecchia approximations with incomplete Cholesky decomposition [Schäfer et al., 2021] within Thompson sampling. Further, correlation-based Vecchia GPs [Kang and Katzfuss, 2021] may be well suited for neural architecture search [Kandasamy et al., 2018].

References

[Balandat et al., 2020] Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*.

[Calandra et al., 2016] Calandra, R., Seyfarth, A., Peters, J., and Deisenroth, M. P. (2016). Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23.

[Cao and Fleet, 2014] Cao, Y. and Fleet, D. J. (2014). Generalized product of experts for automatic and principled fusion of gaussian process predictions. *arXiv preprint arXiv:1410.7827*.

[Chapelle and Li, 2011] Chapelle, O. and Li, L. (2011). An Empirical Evaluation of Thompson Sampling. *Advances in Neural Information Processing Systems*, 24.

[Eriksson and Jankowiak, 2021] Eriksson, D. and Jankowiak, M. (2021). High-Dimensional Bayesian Optimization with Sparse Axis-Aligned Subspaces. *arXiv preprint arXiv:2103.00349*.

[Eriksson et al., 2019] Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. (2019). Scalable Global Optimization via Local Bayesian Optimization. In *Advances in Neural Information Processing Systems*, pages 5496–5507.

[Gardner et al., 2018] Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*.

[Gómez-Bombarelli et al., 2018] Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276.

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

[Hensman et al., 2013] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian Processes for Big Data.

[Jegou et al., 2010] Jegou, H., Douze, M., and Schmid, C. (2010). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.

[Johnson et al., 2017] Johnson, J., Douze, M., and Jégou, H. (2017). Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

[Kandasamy et al., 2018] Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., and Xing, E. (2018). Neural architecture search with bayesian optimisation and optimal transport. *arXiv preprint arXiv:1802.07191*.

[Kandasamy et al., 2015] Kandasamy, K., Schneider, J., and Póczos, B. (2015). High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pages 295–304. PMLR.

[Kang and Katzfuss, 2021] Kang, M. and Katzfuss, M. (2021). Correlation-based sparse inverse Cholesky factorization for fast Gaussian-process inference. *arXiv:2112.14591*.

[Katzfuss and Guinness, 2021] Katzfuss, M. and Guinness, J. (2021). A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, 36(1):124–141.

[Katzfuss et al., 2020] Katzfuss, M., Guinness, J., Gong, W., and Zilber, D. (2020). Vecchia approximations of Gaussian-process predictions. *Journal of Agricultural, Biological, and Environmental Statistics*, 25(3):383–414.

[Katzfuss et al., 2022] Katzfuss, M., Guinness, J., and Lawrence, E. (2022). Scaled Vecchia approximation for fast computer-model emulation. *SIAM/ASA Journal on Uncertainty Quantification*, accepted.

[Letham et al., 2019] Letham, B., Karrer, B., Ottoni, G., and Bakshy, E. (2019). Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519.

[Liu and Liu, 2019] Liu, L. and Liu, L. (2019). Amortized variational inference with graph convolutional networks for gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2291–2300. PMLR.

[Maddox et al., 2021] Maddox, W. J., Stanton, S., and Wilson, A. G. (2021). Conditioning Sparse Variational Gaussian Processes for Online Decision-making. *arXiv preprint arXiv:2110.15172*.

[McIntire et al., 2016] McIntire, M., Ratner, D., and Ermon, S. (2016). Sparse Gaussian Processes for Bayesian Optimization. In *UAI*.

[Mockus, 1989] Mockus, J. (1989). *Bayesian Approach to Global Optimization*. Kluwer, Dordrecht, NL.

[Molga and Smutnicki, 2005] Molga, M. and Smutnicki, C. (2005). Test functions for optimization needs. *Test functions for optimization needs*, 101:48.

[Neal and others, 2011] Neal, R. M. and others (2011). MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

[Schäfer et al., 2021] Schäfer, F., Katzfuss, M., and Owhadi, H. (2021). Sparse Cholesky factorization by Kullback-Leibler minimization. *SIAM Journal on Scientific Computing*, 43(3):A2019–A2046.

[Schürch et al., 2023] Schürch, M., Azzimonti, D., Benavoli, A., and Zaffalon, M. (2023). Correlated product of experts for sparse gaussian process regression. *Machine Learning*, pages 1–22.

[Snoek et al., 2014] Snoek, J., Swersky, K., Zemel, R., and Adams, R. (2014). Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, pages 1674–1682. PMLR.

[Stein, 2014] Stein, M. L. (2014). Limitations on low rank approximations for covariance matrices of spatial data. *Spatial Statistics*, 8:1–19.

[Swersky et al., 2013] Swersky, K., Snoek, J., and Adams, R. P. (2013). Multi-Task Bayesian Optimization. Technical report.

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

[Tran et al., 2021] Tran, G.-L., Milios, D., Michiardi, P., and Filippone, M. (2021). Sparse within sparse gaussian processes using neighbor information. In *International Conference on Machine Learning*, pages 10369–10378. PMLR.

[Tresp, 2000] Tresp, V. (2000). Mixtures of gaussian processes. *Advances in neural information processing systems*, 13.

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

[Wang et al., 2016] Wang, J., Clark, S. C., Liu, E., and Frazier, P. I. (2016). Parallel bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*.

[Wang et al., 2018] Wang, Z., Gehring, C., Kohli, P., and Jegelka, S. (2018). Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 745–754. PMLR.

[Wilson et al., 2016] Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR.

[Wilson et al., 2018] Wilson, J. T., Hutter, F., and Deisenroth, M. P. (2018). Maximizing acquisition functions for Bayesian optimization. Technical report.

[Wilson et al., 2017] Wilson, J. T., Moriconi, R., Hutter, F., and Deisenroth, M. P. (2017). The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*.

[Wu and Frazier, 2016] Wu, J. and Frazier, P. I. (2016). The parallel knowledge gradient method for batch Bayesian optimization. In *Advances in Neural Information Processing Systems*.

[Wu et al., 2022] Wu, L., Pleiss, G., and Cunningham, J. (2022). Variational nearest neighbor gaussian processes. *arXiv preprint arXiv:2202.01694*.

[Yuan and Neubauer, 2008] Yuan, C. and Neubauer, C. (2008). Variational mixture of gaussian process experts. *Advances in neural information processing systems*, 21.

A VECHIA AND SVI-GP

The Stochastic Variational Infererence GP (SVI-GP) [Hensman et al., 2013] is an inducing point method that uses stochastic vartiaional inference to train GPs in $\mathcal{O}(m^3)$ time, where m is the number of inducing points. Throughout this paper we compare with SVI-GP so in this section we highlight how Vecchia compares to SVI-GP on a set of simple regression tasks when the function contains sharp minima or high frequency fluctuations. In our examples we let m denote both the number of inducing points for SVI-GP and the number of nearest neighbors for Vecchia. Both models use a Matérn 5/2 kernel whose hyperparameters are tuned using a dataset of where the inputs are generated uniformly over the domain and the input space is rescaled to the unit hypercube. Vecchia uses a random ordering of the data and relies on mini-batch gradient descent during training.

A.1 Michalwicz Function and Sharp Minima

To demonstrate how Vecchia GPs perform on functions with sharp minima, we compared Vecchia GP to SVI-GP on the Michalewicz function [Molga and Smutnicki, 2005].

We created training data by generating $n = 500$ observations from the 15-dimensional version of the function and adding independent noise from $\mathcal{N}(0, 0.05^2)$. We then generated a test set mimicking a typical BO prediction setting by finding the minimum in the training set and generating 50 noiseless realizations in a neighborhood of the minimum.

For a range of values for m, we fitted an exact Gaussian process, the Vecchia approximation and the SVI-GP, and computed all the posteriors on the test set. Then we computed the KL-divergence $KL(p(\boldsymbol{y}_p|\boldsymbol{y}_{1:n})\|\widehat{p}(\boldsymbol{y}_p|\boldsymbol{y}_{1:n}))$ between the exact posterior and each of the approximate posteriors. We repeated this process, including generating new data, and averaged the results as we varied m. For Vecchia, m corresponds to the number of NN, and for SVI-GP, m corresponds to the number of inducing points.

Figure 6 shows the log of the KL-divergence for both models as we increase m. In this example, Vecchia outperformed SVI-GP for all values of m by several orders of magnitude. Based on further exploration, we conjecture that Vecchia more closely matches the local behavior of the function at the optima and SVI-GP is more prone to return estimates similar to the average over the entire domain. This success at modeling local fluctuations makes Vecchia a strong candidate for BO, where we often have sharp spikes at the global optima.

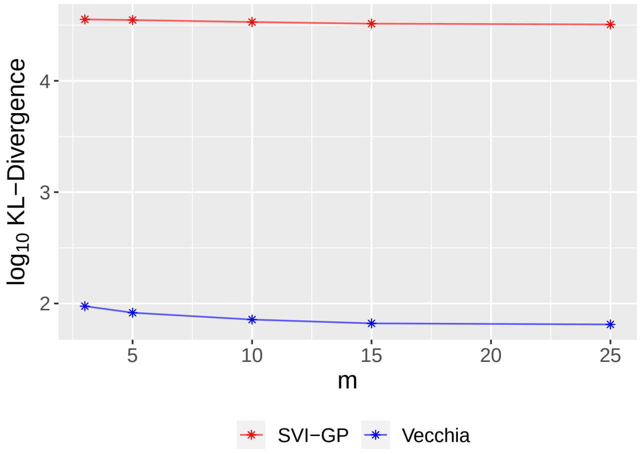

Figure 6: For prediction of the 15-dimensional Michalewicz function, the log of the KL-divergence between an exact GP and approximate GPs based on the Vecchia approximation (blue) and the SVI-GP approximation (red).

A.1.1 Ackley Function and High-Frequency Data

To compare SVI-GP and Vecchia on functions with high-frequency fluctuations, we look at the 1D Ackley function over $[-5, 5]$. Figure 7 shows the means of both SVI-GP and Vecchia along with the true function and the training data. We can see that SVI-GP provides oversmoothed predictions, but matches the low-frequency shape of the function. Vecchia both matches the shape of the function well and captures the high frequency behavior of the function. Vecchia is able to capture this behavior even though the data is ordered randomly. This is good, because it means we can avoid the maximin ordering step and still obtain high-quality results.

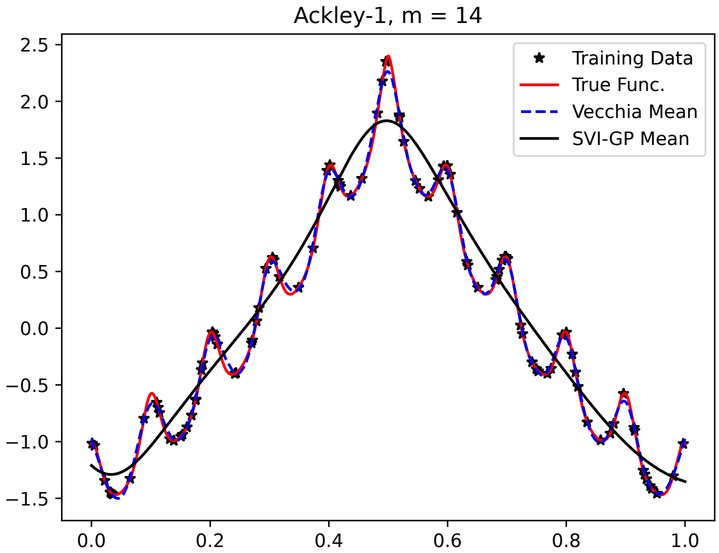

Figure 7: Vecchia (dashed blue line) and SVI-GP (solid black line) posterior means for the 1D Ackley function with the truth (solid red line) and training data (black stars) when $m = 14$.

B APPROXIMATE VARIABLE ORDERING

The function below gives an approximate maxmin ordering which works well empirically, and can be run in parallel. Additionally, we assume there is a function that can perform the exact maxmin ordering, such as Algorithm C.1 in the supplementary material of [Schäfer et al., 2021]. The algorithm works by dividing the inputs into smaller and smaller groups until each group contains less than some predetermined number of points. Then, exact maximin ordering is performed on these groups. Finally, the ordering from each group is combined into the final ordering.

Algorithm 1 Approximate MaxMin Ordering

```
**Input:** data $\mathbf{x}$ (i.e. inputs to target function), parameters $MaxSubsetSize$
$N \leftarrow length(\mathbf{x})$
**if** $N > MaxSubsetSize$ **then**
    $order1 \leftarrow ApproximateMaxMin(\mathbf{x}[0 : floor(N/2)], MaxSubsetSize)$ {Run in Parallel.}
    $order2 \leftarrow ApproximateMaxMin(\mathbf{x}[floor(N/2) : N], MaxSubsetSize)$ {Run in Parallel.}
    $order = concatenate(order1, order2)$
**else**
    $order = GetExactMaxMin(\boldsymbol{x})$
**end if**
Return $order$
```

C VARIANCE CALIBRATION

The algorithm for recalibrating the predictive variance for Vecchia GPs based on holdout set.

The function $KNN(\boldsymbol{a}, \boldsymbol{b}, c)$ returns the c NN of $\boldsymbol{b}$ in $\boldsymbol{c}$ and the function $subsample(\boldsymbol{i}, j)$ selects a subsample of size j from $\boldsymbol{i}$. The index set I_{x^*} are the last q points added to the dataset. That is the last query generated.

Algorithm 2 Vecchia Variance Calibration

```
  Input: data $\mathbf{x}, \mathbf{y}$, parameters $\boldsymbol{\theta}$, $I_x^*$
  $I_O \leftarrow KNN(\mathbf{x}[I_{x^*}], \mathbf{x}, 5 * q)$ {finding neighbors of current best value of $\boldsymbol{x}$.}
  $I_O = subsample(I_O, q)$ {randomly keeping $q$ of the best $\boldsymbol{x}$'s neighbors.}
  $\tilde{X} = \mathbf{x}[I_O] \cup \mathbf{x}[I_{\boldsymbol{x}^*}]$ {forming validation set composed of best $\boldsymbol{x}$ and its $q$ neighbors.}
  $\tilde{y} = \mathbf{x}[I_O] \cup \mathbf{y}[I_{\boldsymbol{x}^*}]$
  $\tilde{\mathcal{D}} = \{\mathbf{x} \setminus \tilde{X}, \mathbf{y} \setminus \tilde{y}\}$ {Create a dataset excluding the validation set.}
  $\mathcal{L}_{b_v} = p(\tilde{y}; \tilde{\boldsymbol{\mu}}_{\boldsymbol{\theta}^*}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}^*} + b_v, \tilde{\mathcal{D}})$ {Form prediction for validation and inflate the posterior variance by $b_v$.}
  Return: $\arg\max_{b_v \in [0,2]} \mathcal{L}_{b_v}$ {Return the $b_v$ that maximized the likelihood of the validation data.}
```

D PREDICTION

We consider the response first ordering (RF) using RF-standard (RF-stand) [Katzfuss et al., 2020]. In RF-stand, the jth prediction location conditions on any previously ordered data point, including another prediction location. This means we take into account the posterior covariance between some prediction locations if they are closer together than to training data. For clarity, we use the subscript r to refer to the observed data and the subscript and superscript p to refer to predictions.

We have

$$\widehat{p}(\boldsymbol{y}^p | \boldsymbol{y}_{1:n}) = \prod_{i=1}^{n_p} p(y_i^p | \boldsymbol{y}^p_{c_p(i)}) = \mathcal{N}(\boldsymbol{\mu}_p, (\boldsymbol{L}^T \boldsymbol{L})^{-1}),$$

where the mean, $\boldsymbol{\mu}_p$, and inverse Cholesky factor, $\boldsymbol{L}$, of the predictive distribution in 3 can be computed as:

$$\boldsymbol{\mu}_p = (\boldsymbol{U}_{p,p}^T)^{-1} (\boldsymbol{U}_{n,p})^T \boldsymbol{y}_{1:n}, \qquad \boldsymbol{L} = \boldsymbol{U}_{p,p}^T$$

The matrices $\boldsymbol{U}_{p,p}$ and $\boldsymbol{U}_{n,p}$ are both sparse with at most m non-zero elements per column. This means the distribution in (3) can be computed in $\mathcal{O}(n_p m^3)$ time.

For prediction we form two matrices $\boldsymbol{U}_{r,p}$ and $\boldsymbol{U}_{p,p}$ whose entries can be computed using equation 11 [Katzfuss et al., 2020]. Using the $\boldsymbol{U}$ matrices we can get the posterior mean and the Cholesky factor of the posterior variance with complexity dependent on the number of prediction locations, and not the sample size of the training data.

The Cholesky factor of the posterior precision for prediction locations is given by:

$$\boldsymbol{L}_p = (\boldsymbol{U}_{p,p}^{-1})^T \tag{9}$$

The posterior predictive mean is given by

$$\boldsymbol{\mu}_p = (\boldsymbol{U}_{p,p}^T)^{-1} \boldsymbol{U}_{r,p}^T \boldsymbol{y}_{1:n}. \tag{10}$$

D.1 U Matrix

Consider the matrix $\mathbf{U}$, whose elements are given by:

$$\mathbf{U}_{j,i} = \begin{cases} d_i^{-1/2}, & i = j \\ -b_i^{(j)} d_i^{-1/2}, & j \in g(i) \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where $\boldsymbol{b}_i^T = C(\boldsymbol{x}_i, \boldsymbol{x}_{g(i)}) C(\boldsymbol{x}_{g(i)}, \boldsymbol{x}_{g(i)})$, $d_i = C(\boldsymbol{x}_i, \boldsymbol{x}_i) - \boldsymbol{b}_i^T C(\boldsymbol{x}_{g(i)}, \boldsymbol{x}_i)$ and $C(\boldsymbol{x}_l, \boldsymbol{x}_k) = K(\boldsymbol{x}_l, \boldsymbol{x}_k) + \mathbb{1}_{k=l} \sigma^2$. When we allow the prediction locations to condition on other prediction locations then there will be non-zero elements in the matrix $\mathbf{U}_{pp}$, and there will be columns with less than m non-zero elements in the matrix $\mathbf{U}_{n,p}$. The significance of this becomes obvious when we consider the posterior mean, since a location may condition on less than m observations. The prediction will instead rely on the value that was predicted at other unknown locations.

This works well when the number of observations is much greater than the number of predictions. If we restrict ourselves to condition only on observed data then we ignore the joint nature of the posterior, but gain speed as all the operations can be done in parallel.

In either case, the matrices $\mathbf{U}_{n,p}$ and $\mathbf{U}_{pp}$ will have at most m non-zero elements per column. This is clear from 11 since row j will have a non-zero for column i if $\boldsymbol{x}_i$ conditions on $\boldsymbol{x}_j$, and by design each $\boldsymbol{x}_i$ conditions on at most m nearest neighbors.

E KERNEL AND WARPING

When the target function cannot be modeled by a stationary GP, it can help to first warp the input space through a non-linear invertible function. To incorporate this tool into the Vecchia framework, we consider a general kernel that allows for warping,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tilde{K}(\|w(\boldsymbol{x}) - w(\boldsymbol{x}')\|), \tag{12}$$

where $\| \cdot \|$ denotes Euclidean distance and $\tilde{K}$ is an isotropic covariance function. We assume the warping function to operate element-wise on each of the d input dimensions, such that $w(\boldsymbol{x}) = \big(w_1(\boldsymbol{x}_1), \ldots, w_d(\boldsymbol{x}_d)\big)^\top$, where $w_j(\boldsymbol{x}_j) = \omega_{\boldsymbol{\phi}_j}(\boldsymbol{x}_j)/\lambda_j$; here, λ_j is the range parameter for the jth input dimension, and $\omega_{\boldsymbol{\phi}}$ is bijective, continuous, and differentiable with respect to $\boldsymbol{\phi}$. For our numerical experiments, we used the Kumarswamy CDF [Balandat et al., 2020], $\omega_{a,b}(\boldsymbol{x}) = 1 - (1 - \boldsymbol{x}^a)^b$; when its parameters are set to $a = b = 1$, (4) becomes the well-known automatic relevance determination (ARD) kernel.

For approximating a GP with a warping kernel, we propose the warped Vecchia approximation. Specifically, in (2), we carry out the maximin ordering of $\boldsymbol{y}_{1:n}$ and the selection of NN conditioning sets $c(i)$ based on Euclidean distance between the corresponding warped inputs, $w(\boldsymbol{x}_1), \ldots, w(\boldsymbol{x}_n)$. For prediction of a new variable y_i^* at $\boldsymbol{x}_i^*$, we also select $c^*(i)$ as the m NNs to $w(\boldsymbol{x}_i^*)$ among $w(\boldsymbol{x}_1), \ldots, w(\boldsymbol{x}_n)$. In the special case of linear warping with $\omega(\boldsymbol{x}) = \boldsymbol{x}$, this warped Vecchia approach is equivalent to the scaled Vecchia approximation for emulating computer experiments [Katzfuss et al., 2022].

Generally speaking, as long as the warping function is injective, then computing the distance between warped points $w(\boldsymbol{x})$ and $w(\boldsymbol{x}')$ is equivalent to using some proper distance metric on the original inputs. For this reason, the warped Vecchia and the scaled Vecchia can be seen as simply performing Vecchia GP regression with a learned distance metric.

F EXPERIMENT DETAILS

Below are further details on the experimental setup.

F.1 Experimental Setup

During every round, each GP surrogate model under consideration was fit using all of the available data. Thompson sampling was used within each TuRBO trust region, with the input locations chosen from a Sobol sequence. We evaluated the GP at $\min(5000, \max(5000, 100d))$ locations q-separate times. We chose the batch of q elements to be the locations with the q highest simulated values within Thompson sampling. For all surrogates, we used a Matérn 5/2 covariance function for $\tilde{K}$ in (4), assuming linear warping (i.e., $a = b = 1$) for all examples except the Levy function. For approximate surrogate models, we chose m as described in Section F.2. The input space for all test functions and problems was mapped to $[0, 1]^d$. All results were averaged over independent replicates, with starting points again chosen using a Sobol sequence.

F.2 Choosing m

For the approximate surrogates, we grow m polylogarithmically with n, $m = C_0 * \log_{10}^2(n)$. To simplify our comparison between surrogate models, we chose a single value of C_0 for all our experiments. To choose this m, we looked at the regret vs number of samples when we used the Vecchia approximation with EI as our acquisition function and compared with the regret curve using the exact GP. When using the Six-Hump camel function we found a C_0 value of around 7.2 is the lowest for which Vecchia tracked the exact solution closely. We excluded the Six-Hump camel function from the comparison of regret curves to put the competing approximations on more even footing.

F.3 Levy 55-20 function

We used a modified version of the Levy-20 function to show the benefit of warping with Vecchia. We call this function Levy 55-20, and it is simply the Levy-20 function with 35 extra irrelevant dimensions. The hyper cube on $[0, 1]^{55}$ was warped using $w(\boldsymbol{x}) = S(4 * \boldsymbol{x} - 1)$ ($S(\cdot)$ is the sigmoid function) before being scaled to $[-5, 5]^{55}$. Only the first 20 dimensions are used to evaluate the Levy-20 function.

G IMPACT OF BATCH SIZE DURING TRAINING

In this section we fit a Vecchia GP to data generated from 25D Ackley function on $[-3, 3]^{25}$. There are 1,500 observations in total and we use a Matérn 5/2 ARD covariance function.

Figure 8 shows the negative log-likelihood vs the log epoch (i.e., the number of passes through the data) during training for different batch sizes. We can see that a batch size of approaches the same minima as all other batch sizes, but does so in many fewer passes through the data. While the loss curve is slightly noisy near the end of training, in practice an early stopping criteria is used to avoid unnecessary training time and unstable behavior.

The results shown are not unique to the Ackley-25 function, and are illustrative of the typical training dynamics for Vecchia GPs trained using mini-batch gradient descent. For of our applications we find that a batch size of 64 offers a good balance between stable training and speed.

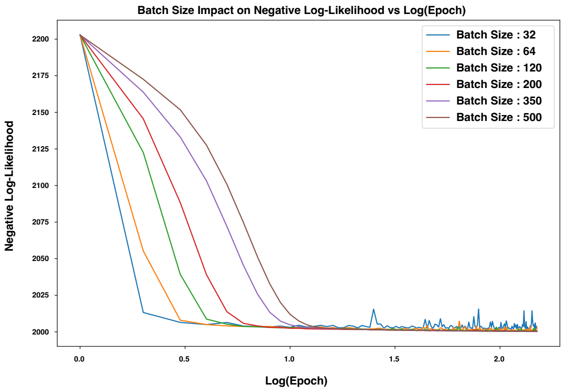

Figure 8: Log-likelihood vs. epoch for different batch sizes.

H NEAREST NEIGHBORS AND ORDERING

The nearest-neighbor search can become a bottleneck in the Vecchia procedure, but we have found that using approximate NN yields good performance. In particular, we suggest using approximate NNs with product quantization [Jegou et al., 2010, Johnson et al., 2017]. This allows for balancing accuracy and speed as we consider how much data is available. As illustrated in Figure 1, in our experience the approximate NN search is often good but not perfect for small to medium maximin index i (middle panel) but it is typically highly accurate for large i.

For ordering data, we have found that an approximate maximin ordering works well. The data is shuffled, split into disjoint sets, ordered within each set, and recombined. Details of this procedure are given in Appendix 1. For really large datasets, maximin ordering can even be replaced by a completely random ordering; while this may result in a slight decrease in accuracy for a given m [Guinness, 2018], it still results in highly competitive predictions (e.g., see Section A.1.1).

Together, these changes result in Vecchia approximations that can be applied to 1,000-dimensional GPs with $n = 100{,}000$ observations and beyond.

I RUNTIME COMPARISON

In this section we look at the runtime performance of the Vecchia GP approximation compared to SVI-GP (GPyTorch implemenation [Gardner et al., 2018]) on two test problems. The first is the the 10-D Michaelwicz (Michaelwicz-10) function on $[0, 2\pi]^{10}$, and the second is the 10-D Ackley function (Ackley-10) on $[-15, 15]^{10}$.

Vecchia's hyperparameters are optimized using Adam with a learning rate of 0.1, and SVI-GP's hyperparameters are optimized using Adam with a learning rate of 0.01. The variational parameters in SVI-GP are optimized using natural gradient descent as suggested in [Hensman et al., 2013]. Both methods are trained using mini-batches of size 32. The general training and evaluation procedure is as follows:

1. Generate $1,500$ uniformly distributed training input/output pairs and 3,000 uniformly distributed test input/output pairs from the target function.
2. Select a value for m (for Vecchia this is number of neighbors and for SVI-GP this is the number of global inducing points).
3. Begin a timer.
4. Fit the GP to the training data. Terminate the training procedure when the relative loss over an exponentially weighted window of mini-batch losses falls below a prespecified threshold.
5. Compute the Logarithmic loss on the test points for the GP posterior.
6. Measure the amount of time that has passed.
7. Repeat this process 10 times.

Both Vecchia and SVI-GP are fit and evaluated for each training test pair. The values of m we use are different for Vecchia and SVI-GP, since SVI-GP requires many inducing points to achieve a comparable level of performance. Vecchia has values of m that range from 2 to 25 on both functions. For SVI-GP we let m vary from 20 to 600 for Ackley-10 and from 50 to 850 on Michaelwicz-10.

The results of this comparison on Michaelwicz-10 are shown in Figure 9, where each point represents the Logarithmic score for a single value of m for one repeat of the experiment. Notice that we are displaying the log of the negative Logarithmic score because we want lower values to correspond to high performance. We can see from Figure 9 that Vecchia consistently outperforms SVI-GP and this is likely due to the quickly fluctuating nature of the Michaelwicz function. For example in 1-D if we limit the function to $[0, 2\pi]$ then the function is smooth and both Vecchia and SVI-GP will perform well, but as we extend the right end of the interval the function becomes more volatile. Our choice of $[0, 2\pi]^{10}$ for this task was meant to highlight that Vecchia can reach a better result faster than SVI-GP when the function is quickly fluctuating.

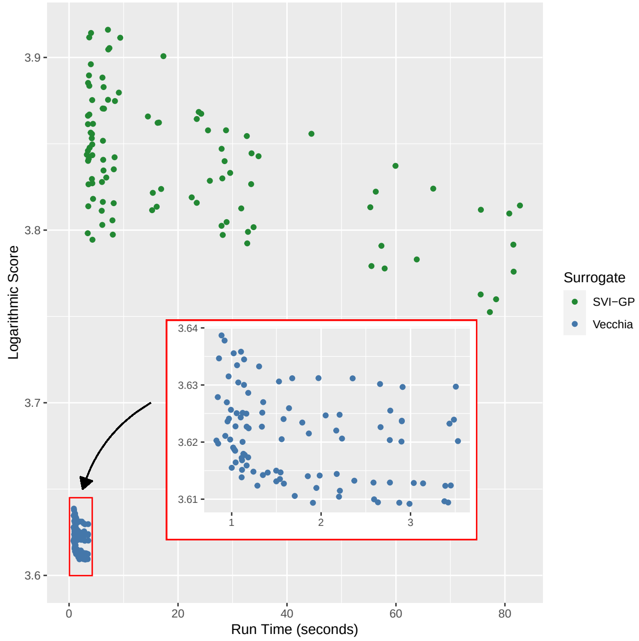

Figure 9: Log of the negative Logarithmic score on test set vs runtime for Vecchia and SVI-GP with the Michaelwicz-10 function. The values of m for Vecchia range from 2 to 25 and for SVI-GP m ranges from 50 to 850. Note: lower values on the y-axis correspond to a better score.

The results on Ackley-10 are shown in Figure 10 where we see that Vecchia outperforms SVI-GP. In Figure 7 we saw that SVI-GP had trouble capturing the location fluctuations on the Ackley-1 function, so it's expected that SVI-GP would perform poorly in high dimensions on this task. We believe a similar type of phenomen is driving the difference in performance. Interestingly, a similar analysis on the Griewank function on [-5,5], which produces a fairly smooth function for such a small domain, showed that Vecchia wasn't improving as much as we increased the value of m, but SVI-GP was marginally increasing even as we approached the upper bound of the m considered. We excluded this result because it was rather bland for both SVI-GP and Vecchia. However, it would be worth exploring which types of GP approximations are amenable to different families of functions. We speculate that SVI-GP and other low rank and inducing point methods will dominate in smooth spaces, but Vecchia is ideal for functions that fluctuate quickly.

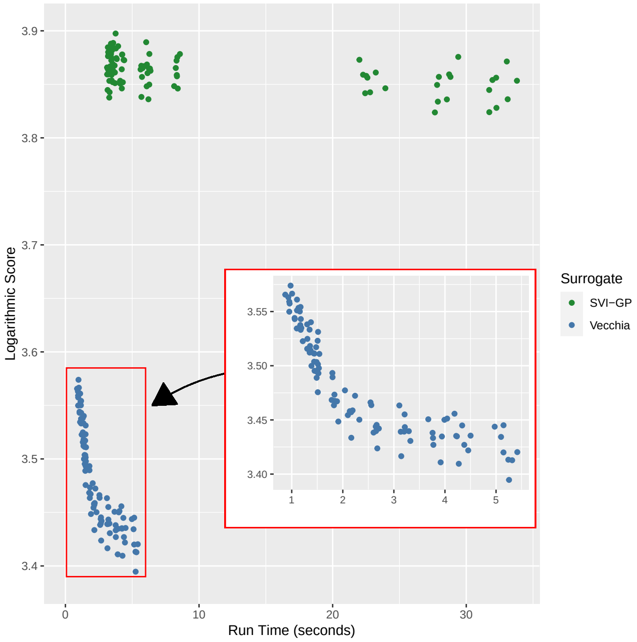

Figure 10: Log of the negative Logarithmic score on test set vs runtime for Vecchia and SVI-GP with the Ackley-10 function. The values of m for Vecchia range from 2 to 25 and for SVI-GP m ranges from 20 to 600. Note: lower values on the y-axis correspond to a better score.

J DETAILS ON MAXIMIN ORDERING

The maximin ordering is most clearly understood by assuming the first location is randomly selected, and then each subsequent location is chosen so that we maximize the minimum distance to all previously ordered locations. To be precise, let i_j denote the order of the i^{th} location, assume $i_1 = 1$ and then select each subsuquent i_j, for j = 2, ..., n such that:

$$i_j = \underset{i \in I \setminus I_{1:j-1}}{\arg\max} \min_{k \in I_{1:j-1}} d(i,k) \tag{13}$$

K RASTRIGIN EXPERIMENTS

Using the Rastrigin function we perform the same types of experiments as in Section 5. The 20 dimensional Rastrigin function is evaluted on $[-5, 10]^{20}$ starting at 50 points and using a batch size (q) of 10. The total budget is 1,000. For the 100 dimensional problem we evaluated the function on $[-5, 10]^{100}$ with 50 initial points, a batch size (q) of 100 and a total budget of 35,000. The results are shown in Figure 11, left corresponds to 20D and right to 100D. For both settings the Vecchia GP outperformed all the surrogates except the exact GP on the 20 dimensional problem. We did not run the exact GP for the 100D problem because run time becomes expensive for $N > 20,000$.

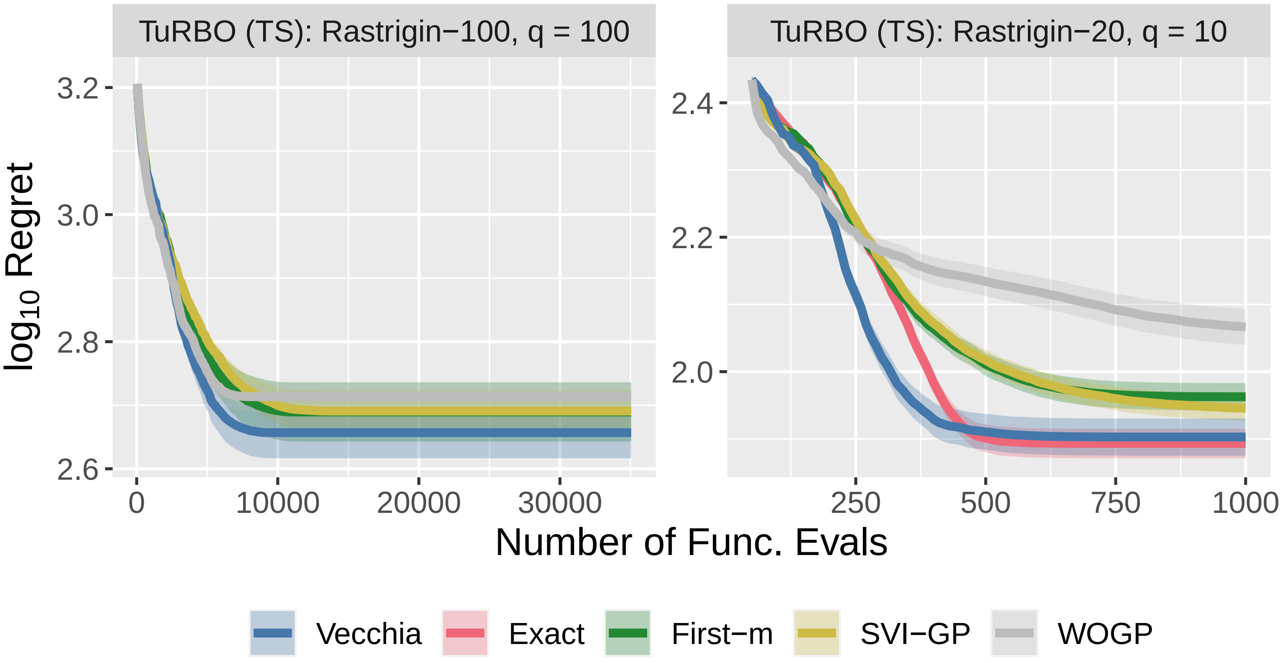

Figure 11: Regret curves for the Rastrigin function in 100 dimensions (left) and 20 dimensions (right). The Vecchia GP outperforms all other approximate surrogates, and is close to the exact when exact is included.

L ABLATION

L.1 Warping

To generate sparse data we take the Levy-2 function on its standard input space and add 8 irrelevant input dimensions. The training data has 15,000 observations and the test data has 500 observations. The inputs points are generated uniformly at random on the 10 dimensional hyper-cube.

L.2 Approximate Nearest Neighbors

We estimate the hyperparameters for the Vecchia GP ($m = 100$) using exact nearest neighbors. Then on 30 indepenent test sets (each of size 1,000) we use approximate nearest neighbors to construct the conditioning sets for predictions. We sweep over values of N_{list} and N_{probe} to construct the conditioning sets and we record the MSE. We also make note of what percentage of the first m exact nearest neighbors we are able to recover for each setting of N_{list} and N_{probe}. The right plot in Figure 5 shows the MSE as we increase N_{probe}, and each curve represents a different value of N_{list}. When N_{probe} is equal to N_{list} the search is exact. We see that for each value of N_{list} the MSE stabilizes to a fixed value as we increase N_{probe}. The dots on each curve represent the value of N_{probe} necessary for at least 70% of the first m nearest neighbors to be correctly recovered.

L.3 Effective Dimension

In this section we examine how the latent dimension of a problem impacts Vecchia performance in terms of MSE. To accomplish this we work with the Ackley function and use the scaled Vecchia GP.

In this problem d_a, the ambient dimension, describes the dimension of the inputs space $\boldsymbol{x} \in \mathbb{R}^{d_a}$. Additionally, d_e, the effective dimension, describes the true input dimension of the function, f. That is $f : \mathbb{R}^d_e \to \mathbb{R}$ and $d_e < d_a$. For this problem we let $d_a = 50$ and we vary d_e and note the MSE as we change the sample size.

Since Vecchia's conditioning sets are based on Euclidean nearest neighbors, we should expect poor performance for high

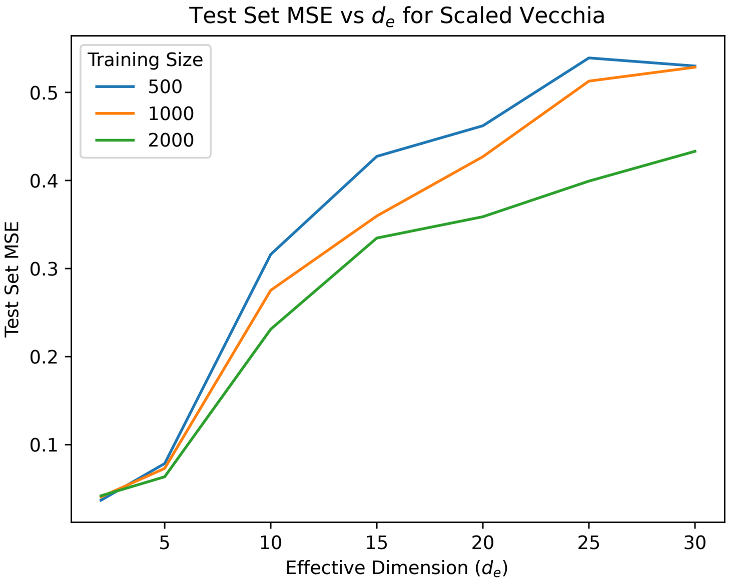

Figure 12: The average MSE vs the latent dimension for the Ackley function. In all instances the ambient dimension is held fixed at 50 ($d_a = 50$), but the latent dimension is varied (d_e). The different color curves represent the number of observations present in the training set. We average over three repeats for every combination of training size and ambient dimension.

effective dimension but good performance with low effective dimension. The reason being, that nearest neighbors in a truly high dimensional space may result in a bad conditioning set, but Vecchia should be able to take advantage of a low effective dimension. This should hold even though the ambient dimension is relatively high ($d_a = 50$) because we compute nearest neighbors in a scaled space (scaled by the learned GP lengthscales).

In Figure 12 we observe that as we increase the sample size the MSE decreases, but for $d_e = 30$ we need a sample size of 2000 to get an average MSE of over 0.4. When $d_e = 2$, every sample size results in an MSE below 0.05. This supports the idea that nearest neighbor based Vecchia is excellent for when the effective dimension is low, but nearest neighbor conditioning sets may not be best for higher effective dimensions. However, these results do not hold if the conditioning sets are chosen in another way, such as a correlation based approach [Kang and Katzfuss, 2021], or for Euclidean nearest neighbors based on warping.

L.4 Variance Calibration

In this section we check how the variance calibration affects the negative log-likelihood (NLL) for the Vecchia and compare this with SVI-GP. We observed in Appendix A that it is difficult for SVI-GP to model high frequency data. Inspired by this observation we would like to verify the assumption that when the MSE of the Vecchia model is lower than SVI-GP's MSE, then the variance calibration for Vecchia will result in a better NLL than the variance calibration for SVI-GP. To test our hypothesis, we work with the Michaelwicz funtion in 1D over the domain $[0, 6\pi]$, with a training set size of 500. The Vecchia GP uses $m = 50$ neighbors and the SVI-GP is given $m = 50$ inducing points. We fit both models to the training data and form predictions at a collection of 500 test points. We then calibrate the variances for each model using the same procedure as in Section 4.4 but with an out of sample validation set. We repeat this procedure three times and average the results.

For Vecchia, the average MSE was 0.4, and the NLLs before and after calibration were, 178 and 0.9, respectively. For SVI-GP the average MSE was 0.7, and the NLLs before and after calibration were 86 and 1.2, respectively. While calibration did improve NLL for both models, the Vecchia GP had a lower NLL after calibration. We believe this is in part due to the difference in MSE between the two models, since no matter how much we change the variance for SVI-GP, the mean estimate will always limit the NLL we can obtain.

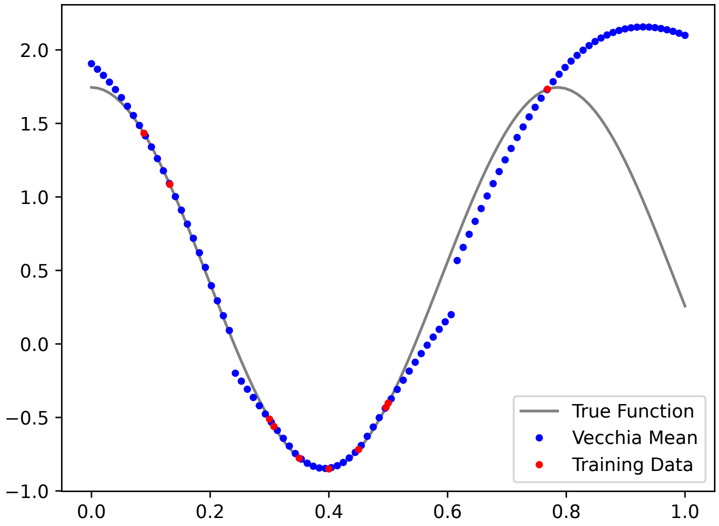

Figure 13: Vecchia ($m = 3$) mean prediction (blue dots) with training data (red dots) and true function (faded black line). We can see that the mean prediction has a noticeable jump at two locations where the conditioning sets change. The region around 0.4 has a higher density of training data and the prediction is much smoother. Comment: dots were used for the mean prediction to highlight the location of the two large jumps, and if an interpolating line were used instead the predictions around 0.4 are still visibly smooth.

M PARALLEL ACQUISITION FUNCTIONS

M.1 Details on qUCB

An acquisition function can be seen as the expectation of an integrand, $\ell(\boldsymbol{y})$, where $\boldsymbol{y} \sim p(\boldsymbol{y}|\mathcal{D}, \boldsymbol{X})$. The integrand $\ell(\boldsymbol{y})$ quantifies how useful it would be to observe a particular $\boldsymbol{y}$. The posterior, $p(\boldsymbol{y}|\mathcal{D}, \boldsymbol{X})$, represents our current belief about how $\boldsymbol{X}$ maps to $\boldsymbol{y}$, given data $\mathcal{D}$.

For example, in this work we use the parallel version of the upper confidence bound (i.e. qUCB). The value of the $qUCB$ acquisition function at location $\boldsymbol{X}$, when $\boldsymbol{y}$ has posterior mean $\boldsymbol{\mu}$ and posterior covariance $\boldsymbol{\Sigma}$, is given by:

$$qUCB(\boldsymbol{X}, \beta) = \int \max(\boldsymbol{\mu} + |\boldsymbol{y} - \boldsymbol{\mu}|)\mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}, \tilde{\boldsymbol{\Sigma}})d\boldsymbol{y}, \tag{14}$$

where $\tilde{\boldsymbol{\Sigma}} := \frac{\beta\pi}{2}\boldsymbol{\Sigma}$ (β is a user specified hyperparameter).

For qUCB the integral in Equation 14 isn't available in closed form, so we must rely on Monte Carlo sampling to estimate the integral [Wilson et al., 2017].

M.2 Vecchia Discontinuities and Acquisition Functions

We now discuss the issue of discontinuitie in Vecchia predictions and how these discontinuities relate to optimizing parallel acquisition functions.

For Vecchia, when making a prediction at a location $\boldsymbol{x}_i \in \mathcal{X}$ we form a conditioning set, $g(i) \subset h(i)$, $|g(i)| = m$, based on the m nearest neighbors. If we move $\boldsymbol{x}_i$ the conditioning set may change. At the point where one nearest neighbor is replaced by another, there will be a discontinuity in both the posterior mean and variance. The collections of points (in 1D), lines (in 2D), etc., where these discontinuities occur will have measure zero. This is visualized for the mean in Figure 13 where we have seven discontinuities with two of them being large. However, when the data is dense, as around 0.4, the prediction is smooth and the discontinuities are not noticeable.

To understand the impact of these discontinuities, consider any point $\boldsymbol{x}_i$ that is at least ϵ away from the location of a discontinuity. Within the ϵ ball centered at $\boldsymbol{x}_i$, the posterior mean and covariance will behave just as an exact GP would if the training data was composed of only the points in the conditioning set $g(i)$. Therefore, the unbiasedness of any Monte Carlo gradient estimate based on an exact GP would hold for a Vecchia GP in the ϵ ball centered around $\boldsymbol{x}_i$.