Vecchia-Approximated Deep Gaussian Processes for Computer Experiments

Annie Sauer, Andrew Cooper, and Robert B. Gramacy
Department of Statistics, Virginia Tech, Blacksburg, VA

ABSTRACT
Deep Gaussian processes (DGPs) upgrade ordinary GPs through functional composition, in which intermediate GP layers warp the original inputs, providing flexibility to model nonstationary dynamics. Two DGP regimes have emerged in recent literature. A “big data” regime, prevalent in machine learning, favors approximate, optimization-based inference for fast, high-fidelity prediction. A “small data” regime, preferred for computer surrogate modeling, deploys posterior integration for enhanced uncertainty quantification (UQ). We aim to bridge this gap by expanding the capabilities of Bayesian DGP posterior inference through the incorporation of the Vecchia approximation, allowing linear computational scaling without compromising accuracy or UQ. We are motivated by surrogate modeling of simulation campaigns with up to 100,000 runs—a size too large for previous fully-Bayesian implementations—and demonstrate prediction and UQ superior to that of “big data” competitors. All methods are implemented in the deepgp package on CRAN.

1. Introduction
Virtualization and simulation increasingly play a fundamental role in the design and study of complex systems that are either impossible or infeasible to experiment with directly. Examples abound in engineering (e.g., Zhang et al. 2015), aeronautics (e.g., Mehta et al. 2014), economics (e.g., Kita, Taniguchi, and Nakajima 2016), and ecology (e.g., Johnson 2008), to name a few. Simulation is “cheaper” than physical experimentation, but not “free.” Associated computational costs often necessitate statistical surrogates, meta-models that furnish accurate predictions and appropriate uncertainty quantification (UQ) from a limited simulation campaign. Surrogates may thus stand-in for novel simulation to support downstream tasks such as calibration (Kennedy and O’Hagan 2001), optimization (Jones, Schonlau, and Welch 1998), and sensitivity analysis (Marrel et al. 2009). Surrogate accuracy, combined with effective UQ, is key to the success of such enterprises.

Gaussian processes (GPs) are a common surrogate modeling choice (Rasmussen and Williams 2005; Gramacy 2020) because they offer both accuracy and UQ in a semi-analytic nonparametric framework. However, inference for GPs requires the evaluation of multivariate normal (MVN) likelihoods, which involves dense matrix decompositions that scale cubically with the size of the training data. Computer simulation campaigns used to be small, but recent advances in hardware, numerical libraries, and STEM training have democratized simulation and led to massively larger campaigns (e.g., Kaufman et al. 2011; Liu and Guillas 2017; Sun et al. 2019; Lin et al. 2021; Marmin and Filippone 2022). The literature has since adapted to this bottleneck by borrowing ideas from machine learning and geo-spatial GP approximation. Examples include sparse kernels (Melkumyan and Ramos 2009), local approximations (Emery 2009; Gramacy and Apley 2015; Cole, Christianson, and Gramacy 2021), inducing points (Quinonero-Candela and Rasmussen 2005; Finley et al. 2009), and random feature expansions (Lázaro-Gredilla et al. 2010). See Heaton et al. (2019) and Liu et al. (2020a) for thorough reviews. Here we are drawn to a family of methods that leverage “Vecchia” approximation (Vecchia 1988), which imposes a structure that generates a sparse Cholesky factorization of the precision matrix (Katzfuss et al. 2020a; Datta 2021). When appropriately scaled or generalized (Stein, Chi, and Welty 2004; Datta et al. 2016; Stroud, Stein, and Lysen 2017; Katzfuss and Guinness 2021), and matched with sparse-matrix and multicore computing facilities, Vecchia-GPs dominate competitors (Katzfuss, Guinness, and Lawrence 2020b) on the frontier of accuracy, UQ, and speed.

Despite their prowess in many settings, GPs (and many of their approximations) are limited by the assumption of stationarity; they are not able to identify spatial changes in input–output regimes or abrupt shifts in dynamics. Here too, myriad remedies have been suggested across disparate literatures. Common themes include partitioning (e.g., Kim, Mallick, and Holmes 2005; Gramacy and Lee 2008; Rushdi et al. 2017; Park and Apley 2018), evolving kernels (Paciorek and Schervish 2003; Picheny and Ginsbourger 2013), and injective warping (Zammit-Mangion et al. 2021). Deep Gaussian processes (DGPs)—originating in geo-spatial communities (Sampson and Guttorp 1992; Schmidt and O’Hagan 2003)
but recently popularized by Damianou and Lawrence (2013) with analogy to deep neural networks—address this stationary limitation by layering GPs as functional compositions. Inputs are fed through intermediate Gaussian layers, which act as “warped” versions of the original inputs, before reaching the response. The structure of a DGP is equivalent to that of a “linked GP” (Ming and Guillias 2021), except that middle layers remain unobserved. When the data generating mechanism is nonstationary, DGPs offer great promise. When regimes change abruptly, DGPs mimic partition schemes. When dynamics change more smoothly, they mimic kernel evolution. When stationary, they gracefully revert to ordinary GP dynamics via identity warplings.

Or at least that’s the sales pitch. In practice things are more murky because nonstationary flexibility is intimately twinned with training data size and structure. Unless the experiment can be designed to squarely target locations of regime change (Sauer, Gramacy, and Higdon 2022), a large training dataset is needed before these dynamics can be identified. Cubic scaling in flops is present in multitude, with large dense matrices at each latent DGP layer. Moreover, the unobserved/latent layers pose a challenge for statistical inference as they may not be analytically marginalized a posteriori. Markov chain Monte Carlo (MCMC) sampling (Ming, Williamson, and Guillias 2021; Sauer, Gramacy, and Higdon 2022) exacerbates cubic bottlenecks and limits training data sizes to the hundreds. Approximate variational inference (VI) offers a thrifter alternative to full posterior integration (Damianou and Lawrence 2013; Salimbeni and Deisenroth 2017), but at the expense of UQ. VI for DGPs does not in- and-of itself circumvent cubic expense, but, when coupled with inducing points and mini-batching (Ding, Tuo, and Shahrampour 2021), can be scaled up; allow us to defer a thorough review to Section 4. We find these approaches to be ill-suited to our computer surrogate modeling setting, in which signal-to-noise ratios are high and UQ is a must. We speculate this is because those libraries target benchmark machine learning examples for classification, or for regression settings with high noise.

In this work we expand the utility and reach of fully Bayesian posterior inference for DGP surrogate models through Vecchia approximation at each Gaussian layer. We explore strategic updating of the so-called “neighborhood sets” involved, based on the warplings of intermediate layers. With careful consideration of the myriad specifications in this hybrid context, we demonstrate that a Vecchia-DGP need not sacrifice predictive power nor UQ compared to a full (un-approximated) analog. We also show that it outperforms approximate DGP competitors in both accuracy and UQ on data sizes up to 100,000. An open-source implementation is provided in the deepgp package on CRAN (Sauer 2022).

The remainder of the article is organized as follows. Section 2 establishes notation while reviewing GPs, DGPs, and Vecchia approximation. Section 3 details our Vecchia-DGP framework. Section 4 discusses implementation and reviews related work on/software for large scale DGP inference with an eye toward contrast and to set up benchmark exercises presented in Section 5. That section concludes with a study of a real-world satellite drag computer experiment. A brief discussion is offered in Section 6.

2. Review of Major Themes

2.1. Gaussian Processes: Shallow and Deep

Let \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) represent a (possibly noisy) black-box function, say to abstract a computer model simulation. Consider inputs \( X = (x_1, \ldots, x_n) \) of size \( n \times d \) and corresponding outputs/observations \( Y = f(X) \) of size \( n \times 1 \). Throughout we use lowercase \( x_i \) to refer to the \( i \)th (transposed) row of \( X \), and likewise for \( Y \). Generic GP regression assumes a MVN over the response, \( Y \sim N_n(\mu(X), \Sigma(X)) \). We specify \( \mu(X) = 0 \), as is common in surrogate modeling (Gramacy 2013), yielding the likelihood

\[
\mathcal{L}(Y \mid X) \propto |\Sigma(X)|^{-1/2} \exp \left( -\frac{1}{2} Y^T \Sigma(X)^{-1} Y \right). \tag{1}
\]

Above “\( \propto \)” indicates a dropped multiplicative constant. All of the action is in \( \Sigma(X) \), which is usually specified pairwise as a function of Euclidean distance between rows of \( X \):

\[
\Sigma(X)_{ij} = \tau^2 \left( \frac{||x_i - x_j||^2}{\theta} + g_{i=j} \right). \tag{2}
\]

Any choice of kernel \( k(\cdot) \) leading to positive definite \( \Sigma(X) \) is valid. Most often these are exponentially decreasing in their argument. Popular choices include the squared exponential and Matérn (Stein 1999), but our work here is not kernel specific. Hyperparameters \( \tau^2, \theta, \) and \( g \) govern the scale, lengthscale, and noise, respectively, working together to describe signal-to-noise relationships.

There are many variations on this theme. For example, vectorized \( \theta \) allow the rate of decay of correlation to vary with input direction. Such embellishments implement a form of anisotropy, a term preferred in geo-spatial contexts, or automatic relevance determination in machine learning (Liu et al. 2020). We do not need such modifications in our DGP setup; once latent layers are involved, such flexibility manifests more parsimoniously via those values. Several of our competitors (Section 5) do use additional hyperparameters and/or equivalent affine pre-scaling (Wycoff, Binois, and Gramacy 2021).

Settings of \( (\tau^2, \theta, g) \) may be fitted through the likelihood (1), now viewing \( (X, Y) \) as training data. While derivative-based maximization is possible with numerical solvers, we prefer full posterior inference. We usually fix \( g \) at a small constant, for example, \( g = 10^{-6} \), as is appropriate for deterministic (or very low noise) computer model simulations. However, we don’t see this as a limitation of our contribution, and our software allows for this parameter to be inferred if needed. Further discussion is reserved for Section 6. Regardless of these choices, evaluation of the likelihood (1) relies on both the inverse and determinant of \( \Sigma(X) \). For a dense \( n \times n \) matrix, this is an \( \mathcal{O}(n^2) \) operation with conventional libraries.

Conditioned on training data and covariance formulation \( \Sigma(\cdot) \), predictions at testing locations \( \mathcal{X'} \) (of size \( n_p \times d \)) follow Equation (3) after extending \( \Sigma(\mathcal{X}', \mathcal{X}) \) to cover rows of \( \mathcal{X'} \) paired with \( X \) following Equation (2):

\[
Y \mid Y, X \sim N_{n_p} \left( \mu'(\mathcal{X'}), \Sigma'(\mathcal{X'}, \mathcal{X}) \right)
\]

where

\[
\begin{align*}
\mu' &= \Sigma'(\mathcal{X}', \mathcal{X}) \Sigma(X)^{-1} Y \\
\Sigma' &= \Sigma(\mathcal{X}') - \Sigma(\mathcal{X}', \mathcal{X}) \Sigma(X)^{-1} \Sigma(X, \mathcal{X}).
\end{align*}
\]
This closed-form is convenient but requires the inverse of $\Sigma(X)$, or at least a clever linear solve.

The above GP specification is stationary because only relative positions of training and testing inputs are involved (2). Consequently, identical input–output dynamics apply everywhere in the input space, which can be limiting for some computer simulations. A great example comes from aeronautics/fluid dynamics. Lift forces on aircraft are fundamentally different at high speed versus low, and the transition at the sound barrier is abrupt (Pamadi et al. 2004). Several strategies to relax at high speed versus low, and the transition at the sound simulations. A great example comes from aeronautics/fluid subsequently, identical input–output dynamics apply everywhere or at least a clever linear solve.

The above GP specification is stationary because only relative positions of training and testing inputs are involved (2). Consequently, identical input–output dynamics apply everywhere in the input space, which can be limiting for some computer simulations. A great example comes from aeronautics/fluid dynamics. Lift forces on aircraft are fundamentally different at high speed versus low, and the transition at the sound barrier is abrupt (Pamadi et al. 2004). Several strategies to relax at high speed versus low, and the transition at the sound simulations. A great example comes from aeronautics/fluid subsequently, identical input–output dynamics apply everywhere or at least a clever linear solve.

A two-layer DGP with latent layer $W$ is modeled as

$$
\begin{align*}
Y | W & \sim \mathcal{N}_n(0, \Sigma(W)), \\
W_k & \sim \mathcal{N}_n(0, \Sigma_k(X)), \quad k = 1, \ldots, p.
\end{align*}
$$

(4)

$W = [W_1, \ldots, W_p]$ is an $n \times p$ matrix with each row representing one observation and each column representing a dimension of the latent space. We adopt the deep learning convention of referring to the component dimensions of $W$ as “nodes.” Each node has its own $\Sigma_k(X)$ which may or may not share features, like kernels and hyperparameters, with others. In our setting we specify unit scale and zero noise on latent layers (i.e., $\tau^2 = 1$ and $g = 0$ within $\Sigma_k(X)$) in order to preserve parsimony and identifiability. It is common to fix $p = d$, but autoencoding setups which “squeeze through” lower-dimensional latent layers ($p \ll d$) are not uncommon in high-dimensional settings (Domingues et al. 2018). Again, our work and implementation are not limited to this choice.

Inference for DGPs, including for hyperparameters buried in the $\Sigma$s, requires integration over $W$:

$$
\mathcal{L}(Y | X) = \int \mathcal{L}(Y | W) \prod_{k=1}^{p} \mathcal{L}(W_k | X) \, dW,
$$

(5)

with $\mathcal{L}(Y | W)$ and $\mathcal{L}(W_k | X)$ following Equation (1), or slight variations thereupon. However, this integral is not analytically tractable. The prevailing inferential tools thus rely either on approximate variational methods (Damianou and Lawrence 2013; Salimbeni and Deisenroth 2017) or sampling (Dunlop et al. 2018; Havasi et al. 2018; Ming, Williamson, and Guillas 2021).

Our preferred inferential scheme, detailed in Sauer, Gramacy, and Higdon (2022), prioritizes UQ through a fully-Bayesian MCMC algorithm, hinging on elliptical slice sampling (ESS; Murray, Adams, and MacKay 2010) of latent layers. ESS is specifically designed for sampling variables with MVN priors, is notably free of tuning parameters, and works well in DGP settings (also see Ming, Williamson, and Guillas 2021). We further embrace a Bayesian treatment of kernel hyperparameters: marginalizing $\tau^2$ from the posterior under a reference prior (Gramacy 2020, chap. 5) and adopting Metropolis-Hasting (MH; Hastings 1970) sampling of $\theta$’s (each $\Sigma(W)$ and $\Sigma_k(X)$ with unique lengthscale), all wrapped in a Gibbs scheme. This approach is thorough but computationally demanding. Each Gibbs iteration requires many evaluations of Gaussian likelihoods (1), and several thousands of iterations are needed to burn-in and then explore the posterior.

As an example of DGP warping, consider the two-dimensional “G-function,” featured later in Section 5.1. A visual is provided in the left panel of Figure 1. This function is characterized by inclines and abrupt shifts. A stationary “shallow” GP is unable to distinguish between the steep sloping regions in the corners and the valley regions that form a cross shape in the center. Figure 1 also shows a posterior sample (i.e., a burned-in ESS draw) of the hidden layer $W$ plotted two ways. The middle panels show each of two “nodes” as a function of $X$. Each node is a conditionally independent stationary GP. These nodes depart from the identity mapping (i.e., $W_1 = X_1$ and $W_2 = X_2$) by stretching the inputs along the diagonal. The right panel shows the combined “warping” effect that both nodes have on the evenly gridded $X$. Since only pair-wise distances feed into the final layer (through $\Sigma(W)$), the warping effect is invariant to rotations and translations. ESS samples of $W$ that “bounce” between mirror images of themselves are taken as an indication that the chain is mixing well. In Section 5.1 we show that the warping(s) in the figure accommodate the G-function’s nonstationary dynamics better than many other methods.

2.2. Vecchia Approximation

The Vecchia approximation (Vecchia 1988) is motivated by computational bottlenecks in GP regression, which are compounded in a DGP setting. The underlying idea is basic: any joint distribution can be factored into a product of conditionals $p(y) = p(y_1)p(y_2 | y_1)p(y_3 | y_2, y_1) \cdots p(y_n | y_{n-1}, \ldots, y_1)$. This is true up to any re-indexing of the $y_j$’s. In particular, and to establish some notation for later, any joint likelihood (1) may be factored into a product of univariate likelihoods

$$
\mathcal{L}(Y) = \prod_{i=1}^{n} \mathcal{L}(y_i | y_{c(i)})
$$

(6)

where $c(1) = \emptyset$ and $c(i) = \{1, 2, \ldots, i - 1\}$ for $i = 2, \ldots, n$. The Vecchia approximation instead takes a subset, $c(i) \subset \{1, 2, \ldots, i - 1\}$, of size $|c(i)| = \min(m, i - 1)$. When $m < n$, the strict equality of Equation (6) is technically an approximation, yet we will use equality notation throughout when speaking of the general case with unspecified $m$. This approximation is indexing-dependent for fixed $m < n$, but hold that thought for a moment. Crucially Equation (6), in the context of Equations (1) and (3), induces a sparse precision matrix: $Q(X) = \Sigma(X)^{-1}$. The $(i,j)$th element of $Q(X)$ is 0 if and only if $y_i$ and $y_j$ are conditionally independent (i.e., $i \notin c(j)$ and $j \notin c(i)$). The Cholesky decomposition of the precision matrix, $U_c$ for $Q(X) = U_c U_c^\top$, is even sparser with fewer than $m$ off-diagonal
nonzero entries in each row. We follow Katzfuss et al. (2020a) in working with the upper triangular $U_x$, referred to as the "upper-lower" Cholesky decomposition.

A GP-Vechia approximation requires two choices: an ordering of the data and selection of conditioning sets $c(i)$. There are many orderings that work well (Stein, Chi, and Welty 2004; Guinness 2018; Katzfuss and Guinness 2021), but a simple random ordering is common (Datta et al. 2016; Stroud, Stein, and Lysen 2017; Wu, Pleiss, and Cunningham 2022). The prevailing choice for conditioning sets is "nearest-neighbors" (NN) in which $c(i)$ comprises of integers indexing the closest observations to $x_i$ which appear earlier in the ordering. Approximations based on NN are sometimes referred to as NNGPs (Datta et al. 2016). To demonstrate an ordering and NN conditioning set, the left panel of Figure 2 shows a grid of inputs with random ordering (the numbers plotted). [The other panels will be discussed in Section 3.] For point $i = 45$ (triangle), the NN conditioning set of size $m = 10$ is highlighted by circles. These are the points closest to $x_i$ in Euclidean distance, with indices $j < i$ in the ordering. Sets $c(i)$ for all $i = 1, \ldots, 100$ are chosen similarly.

Under a GP, components of Equation (6) are univariate Gaussian, $\mathcal{N}(y_i | Y_{c(i)}) \sim \mathcal{N}_1(\mu_i(X), \sigma_i^2(X))$, where

$$B_i(X) = \Sigma(x_i, X_{c(i)})\Sigma(X_{c(i)})^{-1}, \quad \mu_i(X) = B_i(X)Y_{c(i)}.$$  
$$\sigma_i^2(X) = \Sigma(x_i) - B_i(X)\Sigma(X_{c(i)}, x_i)$$  

and $X_{c(i)}$ is the row-combined matrix of $X$'s rows corresponding to indices $c(i)$. Foreshadowing a DGP application in Section 3, one may define $B_i(W), \mu_i(W)$ and $\sigma_i^2(W)$ identically but with $w/W$ in place of $x/X$. With this representation, we convert a large $n \times n$ matrix inversion ($O(n^3)$) into $n$-many $m \times m$ matrix inversions ($O(m^3)$), a significant improvement if $m \ll n$.

The details of Vecchia-GPs, including numerous options for orderings, conditioning sets, hyperparameterizations, and computational considerations, are spread across multiple works (e.g., Datta et al. 2016; Guinness 2018, 2021; Finley et al. 2019; Katzfuss et al. 2020a; Katzfuss and Guinness 2021; Datta 2021). These specifications, along with software implementations (e.g., Katzfuss et al. 2020c; Finley, Datta, and Banerjee 2020; Guinness, Katzfuss, and Fahmy 2021), can be rather complex. We do not need such hefty machinery in our DGP setup for computer surrogate models. Much of our research effort involved sifting through this literature to determine what is essential and which variations work best for DGP surrogates. As one example, latent layers $W$ and deterministic $Y = f(\cdot)$ use noise free modeling (small/zero nugget $g$), which affords several simplifications. In
particular, we do not follow Datta et al. (2016) and Katzfuss and Guinness (2021) in distinguishing between noisy observations and latent “true” variables. This greatly streamlines development (Section 3) and reduces computational demands.

3. Vecchia-Approximated Deep Gaussian Processes

Here we detail our Vecchia-DGP model, posterior integration, and other implementation considerations. Our focus remains on two-layer models for simplicity; extension to deeper DGPs comes down to iteration. We begin by fixing ordering and conditioning; ideas to tailor these to the DGP context follow later in Section 3.3. Our software implementation (Section 4.1), supports a wider array of options than those enumerated here, including deeper (Vecchia) DGPs and estimating nuggets for smoothing noisy data (Section 6).

3.1. Inferential Building Blocks

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

\[
Y \mid W \sim \mathcal{N}_n\left(0, (U_w U_w^\top)^{-1}\right)
\]

\[
W_k \sim \mathcal{N}_n\left(0, \left((U_x^{(k)})\Sigma_1^{1/2}\right)^{-1}\right), \quad k = 1, \ldots, p,
\]

where \(\Sigma (W)^{-1} = U_w U_w^\top\) and \(\Sigma_k (X)^{-1} = (U_x^{(k)})\Sigma_1^{1/2} U_x^{(k)}\). Each \(W_k\), having its own Gaussian prior, also has its own Vecchia decomposition. Often these \(U_x^{(k)}\) will share conditioning sets but have disparate hyperparameterization, for example, unique lengthscales \(\sigma_x^{(k)}\). When \(m = n\), this formulation is equivalent to Equation (4). When \(m < n\), \(U_w\) and \(U_x^{(k)}\) have induced sparsity.

We aim to conduct full posterior integration for this model, extending Equation (5) to include integration over hyperparameters, but, as in Section 2.1, this is not analytically tractable. Posterior sampling via ESS and MH regarding \(\mathcal{L}(Y \mid W)\) and \(\mathcal{L}(W_k \mid X)\) requires three ingredients: (i) prior sampling, (ii) likelihood evaluation, and (iii) prediction at unobserved inputs. These are detailed here, for the model in Equation (8), in turn. We shall focus on \(\mathcal{L}(Y \mid W)\), but the idea is immediately extendable to \(\mathcal{L}(W_k \mid X)\). Both are GPs, so the details only differ superficially in notation, and with iteration over \(k = 1, \ldots, p\). Ultimately, these “building blocks” tie together to support posterior sampling, with details following in Section 3.2.

(i) Prior. Direct sampling of \(Y^* \sim \mathcal{N}_n(0, (U_w U_w^\top)^{-1})\), often called “conditional simulation,” involves individually drawing \(y_i^* \sim \mathcal{N}_1(B_i(W) Y^*_{c(i)}, \sigma_i^2(W))\) for \(B_i(W)\) and \(\sigma_i^2(W)\) defined with analogy to Equation (7). An important difference, however, is that the application here crucially relies on previously sampled \(Y^*_{c(i)}\), meaning each \(y_i^*\) must be sampled sequentially. With an eye toward parallel implementation (Section 4.1), we instead leverage the sparsity of the upper-triangular \(U_w\). Katzfuss and Guinness (2021, Proposition 1) derived a closed-form solution for populating \(U_w\), with \((j, i)\)th entry

\[
U_{wij}^j = \begin{cases} \frac{1}{\sigma_i(W)} & i = j \\ -\frac{1}{\sigma_j(W)} B_i(W)[\text{index of } j \in c(i)] & j \in c(i) \\ 0 & \text{otherwise} \end{cases}
\]

for \(B_i(W)\) and \(\sigma_i(W)\) in Equation (7). With \(U_w\) in hand, sampling \(Y^*\) follows Gelman et al. (2013, App. A): 

\[
Y^* = (U_w^\top)^{-1} z \quad \text{where} \quad z \sim \mathcal{N}_n(0, \mathbb{I}).
\]

Strategically, we avoid matrix inversions by using a forward solve of \(U_w^\top Y^* = z\).

(ii) Likelihood. Evaluations of \(\mathcal{L}(Y \mid W)\) could similarly be calculated as the product of univariate Gaussian densities, combining Equations (6)–(7) via \(\mathcal{L}(y_i \mid W) \sim \mathcal{N}_1(\mu_i(W), \sigma_i^2(W))\). We instead choose to leverage the sparse \(U_w\) formulation (9), yielding the log-likelihood

\[
\log \mathcal{L}(Y \mid W) \propto \log |U_w U_w^\top|^{-1/2} - \frac{1}{2} Y^\top U_w U_w^\top Y
\]

\[
\approx \sum_{i=1}^n \log(U_{ii}^w) - \frac{1}{2} Y^\top U_w U_w^\top Y,
\]

in which the sparse structure of \(U_w\) allows for thrifty matrix multiplications.

(iii) Prediction. Given observed \(Y \mid W \sim \mathcal{N}_n(0, (U_w U_w^\top)^{-1})\), that is, training observations \(Y\) and a burned-in ESS sample of \(W\), we wish to predict \(Y\) for an \(n_p \times d\) matrix of novel \(W\). Note these novel \(W\) ultimately arise as samples following analogous application of the very same procedure we are about to describe, except for \(W_k\) as the “response” drawn at novel testing sites \(X\) (more on this in Section 3.2). The simplest approach treats each row of \(W\) independently. Independent prediction is sufficient if only point-wise means and variances are required, as is common in many downstream surrogate modeling tasks. For each \(i = 1, \ldots, n_p\), we form \(c(i)\) with \(m\) training locations from \(W\) (details in Section 3.3). This imposes conditional independence among \(Y_i\) (i.e., \(Y_i\) is not conditioned on \(Y_j\) for \(i \neq j\)). The posterior predictive distribution then follows \(Y_i \sim \mathcal{N}_1(\mu_i(W), \sigma_i^2(W))\) with \(\mu_i(W)\) and \(\sigma_i^2(W)\) defined as in Equation (7).

This independent treatment is fast and easily parallelized over index \(i = 1, \ldots, n_p\). Consequently, it is the method we prefer for the benchmarking exercises of Section 5, involving a cumbersome additional layer of Monte Carlo (MC) over training-testing partitions. Yet an imposition of independence among \(Y_i\) can be limiting. In some cases, joint prediction using the full covariance structure \(Y \mid Y, W \sim \mathcal{N}_{n_p}(\mu^*, \Sigma^*)\) is essential. We can accommodate such settings as follows. First append \(Y\) indices to the existing ordering of \(W\), ensuring predictive locations are ordered after training locations, forming the full ordering \(i = 1, \ldots, n, n+1, \ldots, n+n_p\). Conditioning sets \(c(i)\) for \(i = n+1, \ldots, n+p\) index any observations from \(W\) or \(Y\) with indices prior to \(i\) in the combined ordering, thus allowing predictive outputs to potentially condition on other predictive outputs, in addition to nearby training data observations.
Next, “stack” training and testing responses in the usual way (Gramacy 2020, sec. 5.1.1)

\[
\begin{bmatrix}
Y \\
\mathcal{Y}
\end{bmatrix} \sim \mathcal{N}_{n+p} (0, \Sigma_{\text{stack}}) \quad \text{where}
\]

\[
\Sigma_{\text{stack}} = \Sigma \begin{bmatrix}
\mathcal{W} \\
\mathcal{W}
\end{bmatrix} = \begin{bmatrix}
\Sigma(W) & \Sigma(W, \mathcal{W}) \\
\Sigma(W, \mathcal{W}) & \Sigma(\mathcal{W})
\end{bmatrix}.
\]

Then leverage (9) to analogously populate a “stacked” upper-
lower Cholesky decomposition,

\[
U_{\text{stack}} = \begin{bmatrix}
U_w & U_{w, \mathcal{W}} \\
0 & U_{\mathcal{W}}
\end{bmatrix}
\]
such that

\[
\Sigma_{\text{stack}} = (U_{\text{stack}} U_{\text{stack}}^\top)^{-1} = \left( \begin{bmatrix}
U_w U_w^\top + U_{w, \mathcal{W}} U_{w, \mathcal{W}}^\top & U_{w, \mathcal{W}} U_{\mathcal{W}}^\top \\
U_{\mathcal{W}} U_{w, \mathcal{W}}^\top & U_{\mathcal{W}} U_{\mathcal{W}}^\top
\end{bmatrix} \right)^{-1}.
\]

An application of the partition matrix inverse identities (details in supplementary materials C.1–C.2) results in the following conditioning identities for \( \mathcal{Y} \mid Y, W \) (3):

\[
\mathcal{Y} \mid Y, W \sim \mathcal{N}_{n+p} (\mu^\ast, \Sigma^\ast)
\]

for

\[
\mu^\ast = -(U_{\mathcal{W}}^\top)^{-1} U_{w, \mathcal{W}} Y \quad \text{and} \quad \Sigma^\ast = \left( U_{\mathcal{W}} U_{\mathcal{W}}^\top \right)^{-1}.
\]

These are simplified versions of the moments provided by Katz-
fuss et al. (2020a), thanks to a streamlined latent structure and the imposi-
tion that predictive locations must be ordered after training locations. Naturally, if no predictive locations condition on others then \( U_{\mathcal{W}} \) and \( \Sigma^\ast \) will be diagonal, and we can return to the simpler implementation of independent predictions.

Katzfuss et al. (2020a) remark that conditioning on other predictive locations, that is, using joint \( \mu^\ast \) and \( \Sigma^\ast \), is more accurate than conditioning only on training data. Anecdotally, in our own empirical work, we have found this difference to be inconsequential. Unless a joint \( \Sigma^\ast \) is required, say for calibration (Kennedy and O’Hagan 2001), we prefer the faster, parallel-
able, independent approach. (Both are provided by our software; more in Section 4.1.) In settings where it might be desirable to reveal/leverage posterior predictive correlation, but perhaps it is too computationally burdensome to work with \( n_p^2 \) pairs of testing sites simultaneously, a hybrid or batched scheme might be preferred.

### 3.2. Posterior Inference

Building blocks (i–iii) in hand, posterior sampling by MCMC may commence following Algorithm 1 of Sauer, Gramacy, and Higdon (2022). In other words, the underlying inferential framework is unchanged modulo an efficient (Vecchia) method for (a) prior sampling, (b) likelihood evaluation, and (c) prediction. Rather than duplicate details here, allow us point out a few relevant highlights. In model training, every evaluation of a Gaussian likelihood uses Equation (11), whether for inner \( (W_k) \) or outer \( (Y) \), matched with \( U_{x}^{(k)} \) and \( U_w \), respectively, and with appropriate covariance hyperparameters, for example, \( \theta_{i}^{(k)} \), embedded into the \( B_{i}(-) \) and \( \sigma_{i}(-) \) components of said \( U \) matrices (9). When employing ESS for \( W \), say, random samples from the prior follow Equation (10). The MCMC scheme remains unchanged modulo Vecchia approximation everywhere under-the-hood.

To predict at unobserved inputs \( X \), that is, Section 4.1 of Sauer, Gramacy, and Higdon (2022), replace traditional GP prediction at each Gaussian layer (3) with its Vecchia counterpart (12). For each candidate (burned-in/thinned) MCMC iteration, predictive locations \( X \) are mapped to “warped” locations \( \mathcal{W} \), which are then mapped to posterior moments for \( \mathcal{Y} \), with each step following Equation (12). The resulting moments are post-
processed, with ergodic averages yielding the final posterior predictive moments. Sauer, Gramacy, and Higdon (2022) focused on small training and testing sets, so their setup favored samples from a joint predictive distribution analogous to Equation (12). These may be replaced with independent point-wise, and parallel-
ed predictions as described in Section 3.1 in the presence of a large/dense testing set. We remind the reader that a DGP pre-
dictive distribution is not strictly Gaussian, even though it arises as an integral over Gaussians. However, we find that ergodic averages, represented abstractly here by empirical moments \( \tilde{\mu} \) and \( \tilde{\Sigma} + \text{cov}(\mu) \) through the law of total variance, are a sufficient substitute for retaining thousands of high-dimensional MCMC draws of \( \mu^* \) and \( \Sigma^* \), say, or their pointwise analogues.

It is important to briefly acknowledge the substantial computation inherent in this inferential scheme. The requisite MCMC requires thousands of iterations, each of which necessitates multiple likelihood (11) evaluations. Predictions require averaging across these draws, although thinning can reduce this effort. Despite these hefty computational demands, efficient parallelization (described in more detail momentarily), strategic initializations of latent layers, and other sensible pre-
processing yield feasible compute times even with large data sizes. For example, the Vecchia-DGPs of Section 5.2 with \( n = 100,000 \) were fit in 26 hr on a 16-core machine. We aim to show that this investment pays dividends compared to faster GP and DGP alternatives in terms of prediction accuracy and UQ (Section 5).

### 3.3. Ordering and Conditioning

Each Gaussian component of the DGP could have its own ordering and conditioning set, \( c_{i}^{(k)}(i) \) for \( U_{x}^{(k)} \) and \( c_{w}^{(i)}(i) \) for \( U_{w} \) in the two-layer model, in which orderings denoted by \( i \) need not be the same. Since each \( c_{i}^{(k)}(i) \) acts on the same input space, we simplify the approximation by sharing ordering and conditioning sets across \( k = 1, \ldots, p \), resulting in only two orderings and two conditioning sets, \( c_{i}(i) \) and \( c_{w}(j) \).

These choices are not part of the stochastic process describing the data generating mechanism, although that is an interesting possibility we discuss in Section 6. A consequence of this is that once a chain has been initialized under a particular ordering, yielding \( U_{x} \) or \( U_{w} \) up to hyperparameters like \( \theta \) which are included in the hierarchy describing the stochastic process, \( c_{i}(i) \) and \( c_{w}(j) \) must remain fixed throughout the MCMC in order to maintain detailed balance. It occurred to us to try randomizing over orderings from one MCMC iteration to the next, but the chain does not burn in/achieve stationarity. Each change to one of \( c_{i}(i) \) or \( c_{w}(j) \) causes the chain to “jump” somewhere else. Nevertheless, it could be advantageous to customize aspects of a
Vecchia ordering and conditioning dynamically, say based on a DGP fit or other analysis, or to hedge by averaging results from multiple orderings. This is fine with independent chains.

With this in mind, we adopt the following setup. Begin with a random ordering of indices for each DGP layer. This follows the recommendation of Guinness (2018) and mirrors other recent work on NNGPs (Datta et al. 2016; Stroud, Stein, and Lysen 2017; Wu, Pleiss, and Cunningham 2022). Then select conditioning sets based on NN, as eponymous in the NNGP/Vecchia literature. In $X$ space, calculating $c_{w}(i)$ as the min$(m, i−1)$ nearest points (of lower index) is straightforward. These locations are anchored in place by the experimental design. For latent Gaussian layer $W$, NN conditioning sets can be more involved. Since $W$ is unobserved, we start with no working knowledge of the nature of the warping. (Our prior is mean-zero Gaussian under a distance-based covariance structure with unknown hyperparameterization.) A good automatic initialization for the MCMC is to assume no warping (i.e., $W = X$). In that setting, NN for $W$ based on relative Euclidean distance in $X$ space is sensible.

Such a conditioning set is even workable after considerable posterior sampling, whereby $W$ may have diverged from the identity mapping with $X$. We find that in practice each individual nodes’ (i.e., $W_k$) contribution to the overall multidimensional warping for $k = 1, \ldots, p$ is usually convex. As a visual, consider again Figure 2. The right two panels show conditioning sets (both NN in a certain sense) for $W$ arising as a function of $X$ corresponding to the mapping in Figure 1. The locations of the observations (marked by numbers) represent a warping of the original evenly gridded inputs (left panel). The middle panel shows the conditioning set $c_{w}(45)$ that was selected based on NN in $X$ space. Observe that these highlighted points are equivalent to those of the left panel.

Selecting $c_{w}(j)$ based on $X$ is a good starting point, but we envision scope to be more strategic. Given posterior information about $W$, one may wish to update $c_{w}(j)$ in light of that warping. For example, NN on $W$ could be calculated after burn-in and used as the basis of $U_w$ conditioning for a restarted chain. Such an operation could be viewed as a nonlinear extension of sensitivity pre-warping (Wycoff, Binois, and Gramacy 2021), tailored to the Vecchia approximation. The right panel of Figure 2 shows such a reconditioning. There is some precedence for evolving neighborhood sets in this way from the ordinary Vecchia-GP literature. For example, Katzfuss, Guinness, and Lawrence (2020b) use estimated multiple-lengthscale parameters to find NN based on rescaled inputs $X/\sqrt{\theta}$, vectorizing over columns; Kang and Katzfuss (2021) extend that to full kernel/correlation based rescaling. Both are situated in an optimization based inferential apparatus, and the authors describe a careful “epoch-oriented” scheme to circumvent convergence issues, analogous to maintaining detailed balance in MCMC. Our particular re-burn-in instantiation of this idea, described above, represents a natural extension: an affine warping of inputs for NN calculations is upgraded to a nonlinear one via latent Gaussian layers. Yet in our empirical work exploring this idea (Section 5.1), we disappointingly find little additional value realized by the extra effort for DGPs. We speculate in Section 6 that this may be because we haven’t yet encountered any applications demanding highly nonconvex $W$.

A final consideration involves extending orderings and neighborhood sets to testing sites when sampling from the posterior predictive distribution. Here, we again use NN sets to select $c_{w}(i)$ and $c_{w}(j)$ for predictive/testing locations $X'$ (which are mapped to $W$). In $X$ space, NN sets are fixed once for each row of $X'$. In $W$ space, after MCMC sampling (i.e., model “training”), we leverage the learned warpings ($W^{(t)}$ for $t \in \mathcal{T}$) and calculate NNs in that space to maximize the efficacy of the approximation at each location: for each row of $W$ we recalculate “warped” NN sets for each sample $W^{(t)}$. Resulting predictions are combined with expectation taken over all $t \in \mathcal{T}$. This recalculation of $c_{w}(i)$ for each $t$ requires extra effort, but it is not onerous and focuses computation where it is most needed.

### 4. Implementation and Competition

Here we detail our implementation and provide evidence of substantial speedup compared to a full DGP using the same underlying method but without a Vecchia approximation (Sauer, Gramacy, and Higdon 2022). We see this compartmentalization—identical computation modulo sparsity of inverse Cholesky factors—as one of the great advantages of our approach. To explain and contrast, we then transition to a discussion of other DGP variations where we find that engineering choices (for computational efficiency) are far more strongly coupled to modeling ones (for statistical fidelity), which can adversely affect performance in surrogate modeling settings, that is, high-signal/low noise prediction with appropriate UQ.

#### 4.1. Implementation

We provide an open-source implementation as an update to the deepgp package (Sauer 2022) for R on CRAN. Although we embrace a bare-bones approach, our R/C++ implementations of the “building blocks” in Section 3.1 are heavily inspired by the more extensive GPvecchia (Katzfuss et al. 2020c) and GpGp (Guinness, Katzfuss, and Fahmy 2021) packages. Computational speed relies on strategic parallelization and careful consideration of sparse matrices. For example, we use OpenMP pragmas to parallelize the calculation of each row of the sparse $U_w$ and $U_x^{(k)}$ (9). We use RcppArmadillo (Eddelbuettel and Sanderson 2014) in C++ and Matrix (Bates and Maechler 2021) in R to handle sparse matrix calculations, aspects of which are also parallelized (but usually to a lesser degree) under-the-hood.

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

```r
R> fit <- fit_two_layer(x, y, vecchia = TRUE, m = 25, true_g = eps)
R> fit <- predict(fit, x_pred, lite = TRUE, m = 25)
```

The neighborhood size is specified as $m = m = \min\{25, n−1\}$ by default, where choosing $n−1$ results in no approximation, but still uses the Vecchia implementation, which is useful for debugging and benchmarking. Notice that training and predic-
tion accommodate distinct $m$. We have found that $m = 25$ (for both) works well for our exercises here [further investigation in supplementary material D], but it may be advantageous to use larger $m$ for prediction or to scale $m$ with $n$. We additionally allow predict calls to toggle between independent (lite = TRUE) or joint predictions (lite = FALSE).

A two-phase MCMC option, updating orderings and NN conditioning sets based on a burned-in warping (Section 3.3) is supported by re_approx = TRUE to continue, an S3 method automating additional MCMC from the end of a previous chain. The true_g argument is optional. If it is not provided, then a nugget is estimated along with other hyper-parameters. In our experiments later we fix eps = 1e-8 for all but the satellite drag example (Section 5.2) where we follow others in using eps = 1e-4.

To demonstrate computational improvements over the full un-approximated implementation, the left panel of Figure 3 compares the computation time of 1000 such MCMC iterations between the full implementation of Sauer, Gramacy, and Higdon (2022) and our proposed Vecchia-DGP. A 16-core, hyper-threaded, Intel i9 CPU at 3.6GHz, was used to collect these timings. The full, non-Veccchia implementation experiences cubic-in-$n$ costs and is not generally feasible for sample sizes above several hundred. The Vecchia implementation scales linearly-in-$n$, allowing for much larger data sizes. To contrast approaches to prediction outlined in Section 3.1, the right panel of Figure 3 shows computation times for Vecchia-DGPs on larger training data sizes with both independent (diamonds) and joint (stars) schemes. Observe that independent predictions scale linearly in both $n$ and $n_p$, but joint predictions are more costly. Note the change in scale of the y-axes from the left to the right panel.

### 4.2. Competing Methodology and Software

As an alternative to MCMC, others have embraced variational inference (VI) in which the intractable DGP posterior (5) is approximated with a simpler family of distributions, often also Gaussian (Damianou and Lawrence 2013). Inspired by deep neural networks, Bui et al. (2016) proposed an Expectation Propogation (EP) scheme for DGPs, which is closely related to VI. Salimbeni and Deisenroth (2017) broadened previous VI-like approaches for DGPs by allowing intra-layer dependencies, naming their method “doubly stochastic variational inference” (DSVI). These approaches optimize rather than integrate, which requires less work. However, optimization ignores uncertainty; the fidelity of a VI approximation is linked to the choice of variational family, rather than directly to computational effort. More MCMC always improves posterior resolution; more VI does not. Hyperparameters don’t neatly fit into variational families otherwise preferred for latent nodes, so they often get ignored, or their tuning is left to external validation schemes. By contrast, extra Metropolis easily accommodates a few more hyperparameters without hassle.

In order to handle training data sizes ($n$) upwards of hundreds of thousands, VI-based DGPs use inducing points, an umbrella term covering ideas developed separately as pseudo-inputs in machine learning (e.g., Snelson and Ghahramani 2006) and as predictive processes in geostatistics (e.g., Banerjee et al. 2008). Inducing points impose a low-rank kernel structure by measuring distance-based correlations through a smaller subset of $m \ll n$ reference locations or “knots” in $d$-dimensions. Woodbury matrix identities improve decomposition of the implied full $n \times n$ structure from $O(n^3)$ to $O(nm^2)$. Although often framed as an “approximation,” inducing points can represent a fundamental change to the underlying kernel structure. Large $n$ and $m$ small enough to sufficiently speed up calculations can result in low-fidelity or “blurry” GP approximations (Wu, Pleiss, and Cunningham 2022). Moreover, optimizing inducing point placement can be fraught with challenges (e.g., Garton, Niemi, and Carriquiry 2020). DSVI uses $k$-means to place inducing points near clusters of inputs, but computer experiments often deploy space-filling designs which would ensure there are no clusters.

The Vecchia approximation offers an alternative to inducing points without introducing auxiliary quantities. Although it is sometimes cast as a novel modeling framework rather than an approximation (Datta et al. 2016), a key advantage is that it doesn’t fundamentally change the underlying kernel structure—
at least not in the way inducing points do. Rather, it more subtly imposes sparsity in its inverse Cholesky factor. Although Vecchia can be higher on the computational ladder ($O(nm^3)$), it is able to provide good approximations with $m$ much smaller than that required of inducing points without the “blur” or hassle in tuning the locations of $m \times d$ quantities. Wu, Pleiss, and Cunningham (2022) entertain Vecchia in lieu of inducing points for ordinary GPs via VI with favorable results. It may only be a matter of time before Vecchia is deployed with VI for DGPs. We prefer MCMC for its UQ properties.

Other alternatives to inducing points in a VI context have been suggested, including random feature expansion (RFE; Lázaro-Gredilla et al. 2010). Extending RFE from ordinary to DGPs has been the subject of several recent papers (Cutajar et al. 2017; Laradji et al. 2019; Marmin and Filippone 2022), with some success. Others have taken the opposite route, keeping inducing points but swapping out VI for Hamiltonian Monte Carlo (HMC; Betancourt 2017) for DGPs (Havasi et al. 2018). HMC has an advantage over VI in that hyperparameters can easily be subsumed into the inferential apparatus without external validation. We show (Section 5) that this leads to performance gains on predictive accuracy in surrogate modeling settings. Nevertheless, DSVI is generally considered the state-of-the-art for DGPs in machine learning. We think this is largely to do with computation and prowess in classification tasks. DSVI’s inducing point approximation enables mini-batching to massively distribute a stochastic gradient descent. This seems to work well in classification settings where resolution drawbacks are less acute, but our experience (Section 5) suggests this may not extend to the low-noise regression settings encountered in surrogate modeling of computer simulations.

Ultimately, benchmarking against such alternatives comes down to software, as even the best methodological ideas are only as good, in practice, as their implementations. DSVI is neatly packaged in gpflux for Python (Dutordoir et al. 2021), but requires specifying hyperparameters. Default settings were not ideal for our test problems, and we found manual tuning to be cumbersome. The sampling-based HMC implementation is available on the authors’ GitHub page (Havasi et al. 2018). This performs better, we think precisely because of its ability to more automatically tune hyperparameters in an Empirical Bayes/EM fashion, but uncertainty in these is not included in the posterior predictions. RFE software is on the authors’ GitHub page (Cutajar et al. 2017), but relies on the specification of myriad inputs, with few defaults provided. Despite attempts to port example uses to our surrogate modeling setting, we had limited success. We found that results were uniformly inferior to DSVI, and no predictive uncertainties were provided leading us to ultimately drop RFE from further consideration. Finally, EP is available on the authors’ GitHub page (Bui et al. 2016), but was written in Python2 and relies on legacy versions of several dependencies; we were unable to reproduce a suitable environment to try their code.

5. Empirical Results

Here we entertain MC benchmarking exercises on two simulated examples and one real-world computer experiment. Code to reproduce all results (including for competitors) is available on our public GitHub repository: https://bitbucket.org/gramacylab/deepgp-ex/. We include the following comparators:

- **DGP VEC:** our Vecchia-DGP, via deepgp using defaults, Matérn $\nu = 5/2$ kernel, independent predictions, and “warped” conditioning sets. See Section 4.1.
- **DGP FULL:** full, un-approximated analog of DGP VEC, with everything otherwise identical (also via deepgp). This comparator was not feasible for all data sizes.
- **DGP DSVI:** from Salimbeni and Deisenroth (2017), implemented in gpflux, with Matérn $\nu = 5/2$ kernel. We follow package examples in using 100 $k$-means located inducing points. For numerical stability we required $\epsilon_p = 1e-4$, lower bounding the noise parameter.
- **DGP HMC:** from Havasi et al. (2018), again using 100 inducing points. This code only supports a squared exponential kernel and estimates (i.e., does not fix) the noise parameter ($\sigma/\epsilon_p$). We found no easy way to adjust these specifications, so we let them be.
- **GP:** stationary unapproximated Gaussian process following Equation (3) with Matérn $\nu = 5/2$ kernel and anisotropic lengthscales estimated through MLE. This comparator was not feasible for all data sizes.
- **GP SVEC:** scaled Vecchia-GP of Katzfuss, Guinness, and Lawrence (2020b), via GPVechia and GPgP. This is a fast “shallow” GP where kernel hyperparameters are estimated (11) via lengthscale-adjusted conditioning sets. We use $m = 25$, Matérn $\nu = 5/2$ kernel, and independent predictions to match DGP VEC.

All DGP variations are restricted to two layers. Our metrics include out-of-sample root mean squared error (RMSE) and continuous rank probability score (CRPS; Gneiting and Raftery 2007). Definitions are provided in Supplementary material B. Lower is better for both. While RMSE focuses on accuracy of predictive means, CRPS incorporates point-wise predictive variances, thus, providing insight into UQ. Although our DGP VEC is able to provide full predictive covariance, our competitors DGP DSVI, DGP HMC, and GP SVEC cannot. Computation times are reported in Supplementary materials F. Notably, DGP VEC requires the most computation, and GP SVEC requires the least.

5.1. Simulated Examples

**Schaffer Function.** The two-dimensional “fourth” Schaffer function (Supplementary materials A.1) can be found in the Virtual Library of Simulation Experiments (VLSE; Surjanovic and Bingham 2013). We follow the second variation therein using $X \in [-2,2]^2$. The function is characterized by steep curved inclines followed by immediate drops. These quick turns are challenging for stationary GPs, making the Schaffer function an excellent candidate for DGPs. We fit models to Latin hypercube samples (LHS; Mckay, Beckman, and Conover 1979) of training sizes $n \in \{100,500,1000\}$ with fixed noise $g = 10^{-8}$. We use LHS testing sets of size $n_p = 500$.

Results for 20 MC repetitions—all stochastic components from training/testing re-randomized—are displayed in Figure 4.
All variations of our DGP fits outperform competitors by both metrics. Crucially, our Vecchia-DGP (DGP VEC) matches the performance of the unapproximated DGP (DGP FULL). For this example we additionally implemented a “DGP VEC noU” comparator, identical to DGP VEC but without ordering/conditioning sets reset after pre-burn-in. Observe that this additional work is of dubious benefit empirically. Aside from MC variability, DGP VEC matches DGP VEC noU. Going forward we shall drop DGP VEC noU, focusing on our preferred DGP VEC setup, without evidence that results are better or worse and despite the additional (marginal) cost. Additional discussion is deferred to Section 6. Finally, at the outset we expected the stationary GPs (GP and GP SVEC) to perform poorly given the complexity of the response surface, but were surprised to see them holding their own against DGP HMC, and eventually surpassing it with \( n = 1000 \). We suspect this is a consequence of “blurry” inducing point approximations. The Vecchia approximated GP (GP SVEC) matched the performance of the full GP. Further study of the choice of conditioning set size \( m \) for both GPs and DGPs is provided in supplementary material D.

**G-function.** The G-function (Marrel et al. 2009, supplementary materials A.2), also in the VLSE, is defined in arbitrary dimension. We worked with \( d = 2 \) in Figures 1 and 2. Here, we expand to \( d = 4 \). Higher dimensionality raises modeling challenges and demands larger training sets. We fit models to LHS samples of training sizes \( n \in \{3000, 5000, 7000\} \) with fixed noise \( g = 10^{-8} \). LHS testing sets were of size \( n_T = 5000 \). Results for 20 MC repetitions are displayed in Figure 5. Variations upon this setup with stochastic noise and higher input dimension, yielding similar results, are provided in supplementary material E.

Again, our DGP VEC outperforms both deep and shallow competitors. DGP HMC, aided by its ability to estimate hyperparameters, bests DGP DSVI, but neither benefits from additional training data. Their predictive capability appears to saturate; again, we suspect inducing points are to blame. While it is possible to increase the number of inducing points and potentially improve things, this requires adjustments to the source code and, in our experience, yields only marginal improvements before computation becomes prohibitive. GP SVEC is able to adapt to larger training sizes and surpass these deeper models. Our DGP VEC benefits from estimation of hyperparameters, additional depth, and learning from additional training data. While the full GP was not feasible for larger sample sizes, it significantly outperformed its Vecchia approximated GP counterpart but was not able to catch the nonstationary DGP VEC.
5.2. Satellite Drag Simulation

The Test Particle Monte Carlo (TPM) simulator (Mehta et al. 2014) models the bombardment of satellites in low earth orbit by atmospheric particles, returning coefficients of drag based on particle composition and input variables specifying satellite configuration. Researchers at Los Alamos National Lab, who developed TPM, wished to build a surrogate achieving less than 1% prediction error (measured in root mean squared percentage error, RMSPE, Supplementary materials B) with as few runs of the simulator as possible. Sun et al. (2019) used locally approximated GPs to reach the 1% goal with one million training data points. Later, Katzfuss, Guinness, and Lawrence (2020b) achieved lower RMSPE’s using scaled Vecchia-GPs (GP SVEC). Here, we show that our Vecchia-DGP is able to beat the 1% RMSPE benchmark with as few as \( n = 50,000 \) (and can beat it consistently with \( n = 100,000 \)), and provide better UQ than the stationary GP SVEC alternative.

We work specifically with the GRACE satellite, specified by a seven-dimensional input configuration, and a pure Hydrogen atmospheric composition. For training data, we use random samples from Sun et al.’s (2019) one million runs in sizes of \( n \in \{10,000, 50,000, 100,000\} \). We use random out-of-sample testing sets of size \( n_p = 50,000 \) from the complement, and follow Sun et al. (2019) in fixing \( g = 10^{-4} \). TPM simulations are technically stochastic, but the noise is very small.

In light of these large training data sizes and the accompanying computational burden, we make some strategic choices to initialize our DGP models and set up the MCMC for faster burn-in. First, we scale the seven input variables using estimated vectorized length-scales from GP SVEC (e.g., \( \bar{X}_i/\sqrt{n} \)). This “pre-scaling” is common in computer surrogate modeling (e.g., Wycoff, Binois, and Gramacy 2021), and mirrors the “scaled” component of the GP SVEC model. Second, we “seed” our MCMC by first running a long, thoroughly burned-in, set of iterations for one sample of \( n = 10,000 \) and using the burned-in samples from this fit to initialize the chains for the larger datasets. This isn’t necessary in practice but helps reduce the burden of repeated applications in a MC benchmarking context.

Results for 10 MC repetitions are displayed in Figure 6. Observe that DGP DSVI and DGP HMC results are omitted from these plots. Because they were not competitive (each producing RMSPE’s between 30% and 35%), their inclusion would severely expand the y-axis of the plots and render them hard to read. Our DGP VEC consistently outperforms the shallow/stationary GP SVEC and is able to achieve the 1% RMSPE goal with as few as 50,000 training observations. Compared to GP SVEC counterparts (with matched training/testing data), DGP VEC models have lower CRPS in all 30 MC repetitions (represented by three boxplots of 10) and lower RMSPE in 28 of the 30.

6. Discussion

In this work we have extended the capabilities of full posterior inference for deep Gaussian processes (DGPs) through the Vecchia approximation. DGPs offer more flexibility than shallow GPs as they are able to accommodate nonstationarity through the warpings of latent Gaussian layers. But inference is slow owing to cubic matrix decomposition. With a Vecchia approximation at each Gaussian layer, our fully-Bayesian DGP enjoys computational costs scaling linearly-in-\( n \). We demonstrated the superior UQ and predictive power of Vecchia-DGPs over both approximate DGP competitors and stationary GPs.

We envision many opportunities for extension. Most notably, we restricted our simulation studies to those where the noise parameter \( (g) \) was fixed to a small value. While many computer simulations are deterministic, others are increasingly stochastic in nature (Baker et al. 2022) and prompt estimation of this hyperparameter, and possibly others. Our \texttt{deepgp} software is capable of estimating \( g \) through additional Metropolis steps. In the Vecchia context, our implementation incorporates the \( g \) parameter directly into the kernel by adding it to the diagonal of \( \Sigma(x_i) \) and \( \Sigma(X_{(i)}) \) in Equation (7). Thus, \( g \) is subsumed into \( B_i(W) \) and \( \sigma^2_i(W) \) which in turn populate \( U_{W} \) (9). As a demonstration of this capability, we provide a noisy simulated example in supplementary materials E. Results are very similar to those of Figure 5; our Vecchia-DGP outperforms across the board. Some authors separate responses into “true/latent” and “actual/observed” variables and caution against incorporating the noise parameter directly into the kernel (e.g., Datta et al. 2016; Katzfuss and Guinness 2021), but we have not seen any...
drawbacks to our simplified approach in the DGP context. Furthermore, it may be possible to extend our DGP model to accommodate heteroscedastic noise through additional latent Gaussian random variables (Binois, Gramacy, and Ludkovski 2018), although we caution that a model that is too flexible may fall prey to a signal–noise identifiability trap. One remedy is to provide for replicates in the design (Binois et al. 2019).

We restricted our empirical studies to a conditioning set size of \( m = 25 \), after finding limited advantage to larger \( m \). [supplementary material D explores varying \( m \).] This echoes other GP-Vecchia works which have found success with small conditioning sets (e.g., Datta et al. 2016; Stroud, Stein, and Lysen 2017; Katzfuss, Guinness, and Lawrence 2020b; Wu, Pleiss, and Cunningham 2022). However, the size of \( m \) may be more important in larger dimensional problems. It may also be advantageous to use larger \( m \) for prediction, as it is otherwise rather cheaper than training. Similarly, we entertained only two-layer DGPs. Our experience with three-layer DGPs for surrogate modeling is that one of the latent layers settles into a near identity mapping, resulting in a lot more computation for little gain. In some cases, the added flexibility of a three-layer DGP can lead to overfitting and adversely affect predictions/UQ (Sauer, Gramacy, and Higdon 2022). Our test functions and real-data computer simulations may not be nonstationary enough to warrant two or more levels of warping.

Perhaps the most intriguing extension lies in the choice of the Vecchia ordering and conditioning sets. Common practice, as we embraced, involves simply fixing an ordering and conditioning, sometimes informed by the data as when NNs are scaled or warped. Many works have investigated the effects of different ordering/conditioning structures (e.g., Stein, Chi, and Welty 2004; Stroud, Stein, and Lysen 2017; Guinness 2018), yet these analyses have all been post hoc. If we view the ordering and conditioning structure as components of the stochastic (data generating) process, there may be potential to learn these, either through maximization or similar MCMC sampling. In our simulation studies, updating the NN conditioning sets based on learned latent warping did not affect posterior predictions, perhaps suggesting that inference for these would have similar null-effects. Yet we suspect this is because the learned latent warpings we have encountered tend to rotate and stretch inputs, resulting in minimal effects on the proximity of observations. We are continuing to “hunt” for examples where more flexible neighborhoods are valuable.

**Supplementary Materials**

Equations for simulated functions (Schäffer, G-function) and evaluation metrics (RMSE, CRPS). Derivation of Vecchia-approximated posterior predictive moments (12). Investigation of varying conditioning set size (\( m \)). Additional simulations with noise and higher dimension. Computation times for all exercises.

**Disclosure Statement**

The authors report there are no competing interests to declare.

**Funding**

This work was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program under Award Number 000231018.

**References**

Baker, E., Barbillon, P., Fadikar, A., Gramacy, R. B., Herbei, R., Higdon, D., Huang, J., Johnson, L. R., Ma, P., Mondal, A., Pires, B., Sacks, J., and Sokolov, V. (2022), "Analyzing Stochastic Computer Models: A Review with Opportunities," *Statistical Science*, 37, 64–89. [834]

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008), “Gaussian Predictive Process Models for Large Spatial Data Sets,” *Journal of the Royal Statistical Society*, Series B, 70, 825–848. [831]

Bates, D., and Maechler, M. (2021). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.3–4. [830]

Betancourt, M. (2017), “A Conceptual Introduction to Hamiltonian Monte Carlo,” arXiv preprint arXiv:1701.02434. [832]

Binois, M., Gramacy, R. B., and Ludkovski, M. (2018), “Practical Heteroscedastic Gaussian Process Modeling for Large Simulation Experiments,” *Journal of Computational and Graphical Statistics*, 27, 808–821. [835]

Binois, M., Huang, J., Gramacy, R. B., and Ludkovski, M. (2019), “Replication or Exploration? Sequential Design for Stochastic Simulation Experiments,” Technometrics, 65, 191–203. [833]

Bui, T., Hernández-Lobato, D., Hernández-Lobato, J., Li, Y., and Turner, R. (2016), "Deep Gaussian Processes for Regression using Approximate Expectation Propagation," in *International Conference on Machine Learning*, 1472–1481. PMLR. [831, 832]

Cole, D. A., Christianson, R. B., and Gramacy, R. B. (2021), “Locally Induced Gaussian Processes for Large-Scale Simulation Experiments,” *Statistics and Computing*, 31, 1–21. [824]

Cutajar, K., Bonilla, E. V., Michiardi, P., and Filippone, M. (2017), “Random Feature Expansions for Deep Gaussian Processes,” in *International Conference on Machine Learning*, pp. 884–893. PMLR. [832]

Damianou, A., and Lawrence, N. D. (2013), "Deep Gaussian Processes," in *Artificial Intelligence and Statistics*, pp. 207–215. PMLR. [825, 826, 831]

Datta, A. (2021), "Sparse Cholesky Matrices in Spatial Statistics," arXiv preprint arXiv:2102.13299. [824, 8227]

Datta, A., Banerjee, S., Finley, A. O., and Gelfand, A. E. (2016), "Hierarchical Nearest-Neighbor Gaussian Process Models for Large Geostatistical Datasets," *Journal of the American Statistical Association*, 111, 800–812. [824, 827, 828, 830, 831, 834, 835]

Ding, L., Tuo, R., and Shahrampour, S. (2021), "A Sparse Expansion For Deep Gaussian Processes," arXiv preprint arXiv:2112.05888. [825]

Domínguez, R., Michiardi, P., Zhou, J., and Filippone, M. (2018), "Deep Gaussian Process Autoencoders for Novelty Detection," *Machine Learning*, 107, 1363–1383. [826]

Dunlop, M. M., Girolami, M. A., Stuart, A. M., and Teckentrup, A. L. (2018), "How Deep are Deep Gaussian Processes?" *Journal of Machine Learning Research*, 19, 1–46. [826]

Dutordoir, V., Salimbeni, H., Hambo, E., McLeod, J., Leibfried, E., Artemev, A., van der Wilk, M., Hensman, J., Deisenroth, M. P., and John, S. (2021), “GPflux: A Library for Deep Gaussian Processes,” arXiv preprint arXiv:2104.05674. [832]

Edelbrock, D., and Sanderson, C. (2014), "RcppArmadillo: Accelerating R with High-Performance C++ Linear Algebra," *Computational Statistics and Data Analysis*, 71, 1054–1063. [830]

Emery, X. (2009), "The Kriging Update Equations and their Application to the Selection of Neighboring Data," *Computational Geosciences*, 13, 269–280. [824]

Finley, A., Sang, H., Banerjee, S., and Gelfand, A. (2009), "Improving the Performance of Predictive Process Modeling for Large Datasets," *Computational Statistics & Data Analysis*, 53, 2873–2884. [824]

Finley, A. O., Datta, A., and Banerjee, S. (2020), "spNNGP: R package for Nearest Neighbor Gaussian Process Models," *Journal of Statistical Software*, 103. [827]

Finley, A. O., Datta, A., Cook, B. D., Morton, D. C., Andersen, H. E., and Banerjee, S. (2019), "Efficient Algorithms for Bayesian Nearest Neighbor Gaussian Processes," *Journal of Computational and Graphical Statistics*, 28, 401–414. [827]
Rasmussen, C. E., and Williams, C. K. I. (2005), *Gaussian Processes for Machine Learning*, Cambridge, MA: MIT Press. [824]

Rushdi, A., Swiler, L., Phipps, E., D’Elia, M., and Ebeida, M. (2017), “VPS: Voronoi Piecewise Surrogate Models for High-Dimensional Data Fitting,” *International Journal for Uncertainty Quantification*, 7. [824]

Salimbemini, H., and Deisenroth, M. (2017), “Doubly Stochastic Variational Inference for Deep Gaussian Processes,” arXiv preprint arXiv:1705.08933. [825,826,831,832]

Sampson, P. D., and Guttorp, P. (1992), “Nonparametric Estimation of Nonstationary Spatial Covariance Structure,” *Journal of the American Statistical Association*, 87, 108–119. [824]

Sauer, A. (2022), *deepgp: Sequential Design for Deep Gaussian Processes using MCMC*. R package version 1.0.0. [825,830]

Sauer, A., Gramacy, R. B., and Higdon, D. (2022), “Active Learning for Deep Gaussian Process Surrogates,” *Technometrics*, 1–15. [825,826,830,831,835]

Schmidt, A. M., and O’Hagan, A. (2003), “Bayesian Inference for Nonstationary Spatial Covariance Structure via Spatial Deformations,” *Journal of the Royal Statistical Society*, Series B, 65, 743–758. [824]

Stein, M. L. (1999). *Interpolation of spatial data*. Springer-Verlag. [825]

Stein, M. L., Chi, Z., and Welty, L. J. (2004), “Approximating Likelihoods for Large Spatial Data Sets,” *Journal of the Royal Statistical Society*, Series B, 66, 275–296. [824,827,835]

Stroud, J. R., Stein, M. L., and Lysen, S. (2017), “Bayesian and Maximum Likelihood Estimation for Gaussian Processes on an Incomplete Lattice,” *Journal of Computational and Graphical Statistics*, 26, 108–120. [824,827,830,835]

Sun, F., Gramacy, R. B., Haaland, B., Lawrence, E., and Walker, A. (2019), “Emulating Satellite Drag from Large Simulation Experiments,” *SIAM/ASA Journal on Uncertainty Quantification*, 7, 720–759. [824,834]

Surjanovic, S., and Bingham, D. (2013), “Virtual Library of Simulation Experiments: Test Functions and Datasets,” Available at [http://www.sfu.ca/~ssurjano]. [832]

Vecchia, A. V. (1988), “Estimation and Model Identification for Continuous Spatial Processes,” *Journal of the Royal Statistical Society*, Series B, 50, 297–312. [824,826]

Wu, L., Pleiss, G., and Cunningham, J. (2022), “Variational Nearest Neighbor Gaussian Processes,” arXiv preprint arXiv:2202.01694. [827,830,831,832,835]

Wycoff, N., Binois, M., and Gramacy, R. B. (2021), “Sensitivity Prewarping for Local Surrogate Modeling,” arXiv preprint arXiv:2101.06296. [825,830,834]

Zammit-Mangion, A., Ng, T. L. J., Vu, Q., and Filippone, M. (2021), “Deep Compositional Spatial Models,” *Journal of the American Statistical Association*, 1–22. [824]

Zhang, Y., Zhao, H., Hassinger, I., Brinson, L. C., Schadrer, L. S., and Chen, W. (2015), “Microstructure Reconstruction and Structural Equation Modeling for Computational Design of Nanodielectrics,” *Integrating Materials and Manufacturing Innovation*, 4, 209–234. [824]