Supplement to: Fast Emulation, Modular Calibration, and Active Learning for Simulators with Functional Response

Grant Hutchings[a,b,*], Derek Bingham[b], Kellin Rumsey[a], Earl Lawrence[a]

[a] *Statistical Sciences, Los Alamos National Laboratory, NM, United States*
[b] *Department of Statistics, Simon Fraser University, BC, Canada*

SM1. Detailed FlaGP algorithms

Detailed algorithms for the methods presented in the manuscript are provided here. The methods are implemented in the `R` package `FlaGP`. Algorithm 1 in the manuscript outlines emulator fitting and prediction, while Algorithms 2 and 3 describe modular calibration without and with a discrepancy model. The supplemental algorithms below provide sufficient detail to reproduce the main computational steps.

Algorithm 1 Detailed FlaGP emulator fitting and prediction

Require: Input matrix $\boldsymbol{X} \in \mathbb{R}^{M \times d_x}$ and functional-output matrix $\boldsymbol{Z} \in \mathbb{R}^{d_y \times M}$.

Fit the FlaGP emulator

Precomputing 1: Data setup

1: Scale each column of $\boldsymbol{X}$ to the unit interval so that $\boldsymbol{X} \subset [0,1]^{d_x}$.
2: Center each row of $\boldsymbol{Z}$ by subtracting its mean across the M simulator runs. Scale the centered matrix so that the total output variance is one, as in the manuscript.
3: Compute the singular value decomposition

$$\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T.$$

4: Select the number of retained basis components p_η. If d_r is the rth singular value, the cumulative proportion of variance explained by the first p components is

$$\mathrm{PVE}(p) = \frac{\sum_{r=1}^{p} d_r^2}{\sum_{r=1}^{\min(d_y, M)} d_r^2}.$$

A common choice is the smallest p for which $\mathrm{PVE}(p) \geq 0.95$.
5: Let $\boldsymbol{U}_{p_\eta}$, $\boldsymbol{D}_{p_\eta}$, and $\boldsymbol{V}_{p_\eta}$ denote the retained SVD factors. Define

$$\boldsymbol{B} = \frac{\boldsymbol{U}_{p_\eta}\boldsymbol{D}_{p_\eta}}{\sqrt{M}} \qquad \text{and} \qquad \boldsymbol{W}(\boldsymbol{X}) = \sqrt{M}\,\boldsymbol{V}_{p_\eta}^T,$$

*Corresponding author. Phone: +1 505-667-2200. Present address: P.O. Box 1663, Los Alamos, NM 87545.

Email address: `grant.hutchings@lanl.gov` (Grant Hutchings)

so that $\boldsymbol{Z} \approx \boldsymbol{B}\boldsymbol{W}(\boldsymbol{X})$, with $\boldsymbol{B} \in \mathbb{R}^{d_y \times p_\eta}$ and $\boldsymbol{W}(\boldsymbol{X}) \in \mathbb{R}^{p_\eta \times M}$.

Precomputing 2: Estimate emulator hyperparameters

6: *Option 1 (BLHS):* For each row $j = 1, \ldots, p_\eta$ of $\boldsymbol{W}(\boldsymbol{X})$, use `laGP::blhs.loop` with b_{blhs} divisions per input dimension and r_{est} bootstrap replicates. For each replicate, estimate the lengthscale vector by empirical-Bayes MAP estimation. Use the componentwise median across replicates as $\hat{\boldsymbol{l}}_j$.

7: *Option 2 (stratified or random subsets):* For each basis component, draw r_{est} subsets of size m_{est} using stratified or simple random sampling. Use `laGP::newGPsep` and `laGP::mleGPsep` to obtain an empirical-Bayes MAP lengthscale estimate for each subset, and use the componentwise median as $\hat{\boldsymbol{l}}_j$.

8: Increasing b_{blhs} in Option 1 or m_{est} in Option 2 increases the estimation subset size and computational cost. Subject to the available computational budget, choose the subset size and r_{est} large enough to obtain stable median estimates.

Precomputing 3: Transform the inputs

9: For each basis component $j = 1, \ldots, p_\eta$, define

$$\tilde{\boldsymbol{X}}_j = \boldsymbol{X} \oslash \sqrt{\hat{\boldsymbol{l}}_j},$$

where $\oslash$ denotes elementwise division with column-wise recycling. Equivalently,

$$[\tilde{\boldsymbol{X}}_j]_{ik} = \frac{X_{ik}}{\sqrt{\hat{l}_{j,k}}}, \qquad i = 1, \ldots, M, \quad k = 1, \ldots, d_x.$$

After this transformation, the lengthscales used for local prediction are fixed at $\mathbf{1}_{d_x}$.

Predict at a new input $\boldsymbol{x}^*$

10: For each basis component j, compute the transformed prediction input

$$\tilde{\boldsymbol{x}}_j^* = \boldsymbol{x}^* \oslash \sqrt{\hat{\boldsymbol{l}}_j}.$$

11: Use a fast nearest-neighbor search, such as `FNN::get.knnx`, to find the m nearest rows of $\tilde{\boldsymbol{X}}_j$ to $\tilde{\boldsymbol{x}}_j^*$. Denote the local transformed design and associated basis weights by

$$\tilde{\boldsymbol{X}}_{j,m}^* = \{\tilde{\boldsymbol{x}}_{j,1}^*, \ldots, \tilde{\boldsymbol{x}}_{j,m}^*\} \qquad \text{and} \qquad \boldsymbol{w}_{j,m}^*.$$

12: Under the zero-mean GP specification in the manuscript, define the target-to-neighborhood correlation vector and local correlation matrix by

$$[\boldsymbol{p}_j^*]_l = \rho(\tilde{\boldsymbol{x}}_j^*, \tilde{\boldsymbol{x}}_{j,l}^*), \qquad l = 1, \ldots, m,$$

and

$$[\boldsymbol{P}_j^*]_{ll'} = \rho(\tilde{\boldsymbol{x}}_{j,l}^*, \tilde{\boldsymbol{x}}_{j,l'}^*) + g\,\mathbb{1}\{l = l'\}.$$

Let

$$\Psi_j = \boldsymbol{w}_{j,m}^{*T}(\boldsymbol{P}_j^*)^{-1}\boldsymbol{w}_{j,m}^*.$$

The predictive distribution of $w_j(\boldsymbol{x}^*)$ is Student-t with m degrees of freedom, predictive mean

$$\mu_j(\tilde{\boldsymbol{x}}_j^*) = \boldsymbol{p}_j^{*T}(\boldsymbol{P}_j^*)^{-1}\boldsymbol{w}_{j,m}^*,$$

and scale

$$s_j^2(\tilde{\boldsymbol{x}}_j^*) = \frac{\Psi_j}{m}\left[1 - \boldsymbol{p}_j^{*T}(\boldsymbol{P}_j^*)^{-1}\boldsymbol{p}_j^*\right].$$

13: Assemble

$$\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{x}^*) = [\mu_1(\tilde{\boldsymbol{x}}_1^*), \ldots, \mu_{p_\eta}(\tilde{\boldsymbol{x}}_{p_\eta}^*)]^T$$

and

$$\boldsymbol{\Sigma}_{\boldsymbol{w}}(\boldsymbol{x}^*) = \frac{m}{m-2}\,\mathrm{diag}\{s_1^2(\tilde{\boldsymbol{x}}_1^*), \ldots, s_{p_\eta}^2(\tilde{\boldsymbol{x}}_{p_\eta}^*)\}.$$

Then

$$\boldsymbol{\mu}_{\boldsymbol{\eta}}(\boldsymbol{x}^*) = \boldsymbol{B}\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{x}^*), \qquad \boldsymbol{\Sigma}_{\boldsymbol{\eta}}(\boldsymbol{x}^*) = \boldsymbol{B}\boldsymbol{\Sigma}_{\boldsymbol{w}}(\boldsymbol{x}^*)\boldsymbol{B}^T.$$

Alternatively, draw $w_j^{(s)}(\boldsymbol{x}^*)$ independently from the component-specific predictive distributions and form

$$\boldsymbol{\eta}^{(s)}(\boldsymbol{x}^*) = \sum_{j=1}^{p_\eta} \boldsymbol{b}_j w_j^{(s)}(\boldsymbol{x}^*).$$

If the simulator outputs were standardized before fitting, transform predictions back to the original output scale as needed.

Algorithm 2 Detailed modular calibration with the FlaGP emulator

Require: A fitted FlaGP emulator, field control inputs $\boldsymbol{X}_F$, field outputs $\boldsymbol{Y}^F = [\boldsymbol{y}^F(\boldsymbol{x}_1^F), \ldots, \boldsymbol{y}^F(\boldsymbol{x}_n^F)]$, priors for $\boldsymbol{\theta}$ and σ_y^2, and an optional discrepancy basis $\boldsymbol{K} \in \mathbb{R}^{d_y \times p_\delta}$.

1: The combined simulator input contains $d_x + d_t$ dimensions. For each emulator component j, let $\hat{\boldsymbol{l}}_j$ contain the lengthscales for both the control and calibration inputs. Use m_c neighbors for emulator predictions made during calibration.

Propose calibration parameters

2: Given the current transformed parameter vector

$$\boldsymbol{\psi}^{\text{curr}} = [\text{logit}(\boldsymbol{\theta}^{\text{curr}}), \log(\sigma_y^{2,\text{curr}})]^T,$$

propose

$$\boldsymbol{\psi}^{\text{new}} \sim N(\boldsymbol{\psi}^{\text{curr}}, \boldsymbol{\Sigma}_{\text{prop}}),$$

and transform back to obtain the proposed $\boldsymbol{\theta}$ and σ_y^2.

Compute emulator predictive moments

3: For each field observation $i = 1, \ldots, n$, form the combined input

$$\boldsymbol{u}_i = (\boldsymbol{x}_i^F, \boldsymbol{\theta}).$$

For each basis component $j = 1, \ldots, p_\eta$, transform $\boldsymbol{u}_i$ using $\hat{\boldsymbol{l}}_j$, select its m_c nearest neighbors, and apply Algorithm 1 with m replaced by m_c to obtain the predictive mean $\mu_{w,j,i}$ and scale $s_{w,j,i}^2$.

4: Assemble the basis-weight predictive moments

$$\boldsymbol{\mu}_{\boldsymbol{w},i} = [\mu_{w,1,i}, \ldots, \mu_{w,p_\eta,i}]^T,$$

$$\boldsymbol{\Sigma}_{\boldsymbol{w},i} = \frac{m_c}{m_c - 2} \text{diag}(s_{w,1,i}^2, \ldots, s_{w,p_\eta,i}^2),$$

and project them to the functional-response space:

$$\boldsymbol{\mu}_{\boldsymbol{\eta},i} = \boldsymbol{B}\boldsymbol{\mu}_{\boldsymbol{w},i}, \qquad \boldsymbol{\Sigma}_{\boldsymbol{\eta},i} = \boldsymbol{B}\boldsymbol{\Sigma}_{\boldsymbol{w},i}\boldsymbol{B}^T.$$

Fit the optional discrepancy model

5: **if** a discrepancy basis $\boldsymbol{K}$ is included **then**

6: Form the residual matrix using the emulator predictive means,

$$\boldsymbol{R}_y = \boldsymbol{Y}^F - [\boldsymbol{\mu}_{\boldsymbol{\eta},1}, \ldots, \boldsymbol{\mu}_{\boldsymbol{\eta},n}].$$

7: Compute the projected discrepancy weights

$$\boldsymbol{V}^T = (\boldsymbol{K}^T\boldsymbol{K})^{-1}\boldsymbol{K}^T\boldsymbol{R}_y.$$

Fit p_δ independent full zero-mean GP models to the rows of $\boldsymbol{V}^T$, using $\boldsymbol{X}_F$ as the input matrix and, for example, `laGP::newGPsep` and `laGP::mleGPsep`.

8: At each observed control input $\boldsymbol{x}_i^F$, compute the discrepancy-weight predictive mean $\boldsymbol{\mu}_{\boldsymbol{v},i}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{v},i}$. Project them to the functional-response space:

$$\boldsymbol{\mu}_{\boldsymbol{\delta},i} = \boldsymbol{K}\boldsymbol{\mu}_{\boldsymbol{v},i}, \qquad \boldsymbol{\Sigma}_{\boldsymbol{\delta},i} = \boldsymbol{K}\boldsymbol{\Sigma}_{\boldsymbol{v},i}\boldsymbol{K}^T.$$

9: **else**

10: Set $\boldsymbol{\mu}_{\boldsymbol{\delta},i} = \mathbf{0}_{d_y}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\delta},i} = \mathbf{0}_{d_y \times d_y}$ for all i.

11: **end if**

Evaluate the transformed-space log posterior

12: For computational efficiency, approximate the joint calibration likelihood by the product of the observation-specific marginal predictive densities. For each i, define

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{\boldsymbol{\eta},i} + \boldsymbol{\mu}_{\boldsymbol{\delta},i},$$

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{\boldsymbol{\eta},i} + \boldsymbol{\Sigma}_{\boldsymbol{\delta},i} + \sigma_y^2 \boldsymbol{I}_{d_y},$$

and evaluate

$$\boldsymbol{y}^F(\boldsymbol{x}_i^F) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i).$$

The efficient likelihood calculation in Appendix A of the manuscript reduces each observation-specific factorization to dimension p_η without discrepancy, or $p_\eta + p_\delta$ with discrepancy.

13: Evaluate

$$\log \pi(\boldsymbol{\psi} \mid \cdot) = \sum_{i=1}^{n} \log p\{\boldsymbol{y}^F(\boldsymbol{x}_i^F) \mid \boldsymbol{\theta}, \sigma_y^2\} + \log p(\boldsymbol{\theta}) + \log p(\sigma_y^2) + \log |J(\boldsymbol{\psi})|,$$

where the change-of-variables term is

$$\log |J(\boldsymbol{\psi})| = \sum_{k=1}^{d_t} \{\log \theta_k + \log(1 - \theta_k)\} + \log \sigma_y^2.$$

14: Apply the Metropolis–Hastings acceptance rule and repeat the proposal, prediction, optional discrepancy, and posterior-evaluation steps to obtain posterior samples.

SM2. Replacing points in a prediction neighborhood

When a candidate point replaces one member of a fixed-size nearest-neighbor prediction set, prediction error is not guaranteed to decrease, even when the candidate is closer to the prediction location than the point it replaces. This behavior is unsurprising in at least two settings:

- the replacement makes the prediction location an extrapolation point; or
- neighbors are selected under a distance metric that does not reflect the fitted anisotropic GP correlation structure.

The second setting can be illustrated with the scalar response

$$y(x_1, x_2) = x_1.$$

The response varies with x_1 but is constant in x_2. Under the Gaussian correlation parameterization used in the manuscript, one therefore expects a shorter or finite lengthscale in the x_1 direction and a very long lengthscale in the x_2 direction. Input scaling consequently places greater importance on closeness in x_1. Suppose prediction is required at $\boldsymbol{x}_p = (0.5, 0.5)$ using two neighbors, with current neighborhood points $\boldsymbol{x}^1 = (0.4, 0.4)$ and $\boldsymbol{x}^2 = (0.6, 1)$. Replacing $\boldsymbol{x}^2$ by the Euclidean-closer point $\boldsymbol{x}^3 = (0.7, 0.4)$ can increase prediction error because $\boldsymbol{x}^2$ is closer to the target in the important x_1 direction.

Prediction error can also increase when the neighborhood does not induce extrapolation and the distance is isotropic. Consider

$$y(x_1, x_2) = x_1 + x_2,$$

for which both inputs are equally important. Nearest neighbors are selected using Euclidean distance

$$d(\boldsymbol{x}, \boldsymbol{x}') = \left\{ \sum_{j=1}^{2} (x_j - x'_j)^2 \right\}^{1/2}.$$

This ordering is equivalent to that induced by an isotropic Gaussian correlation function, which is monotone in squared Euclidean distance.

Figure SM1 summarizes the example. The red $\times$ marks the prediction location, the black circles are the initial training points, and the blue points are the five current nearest neighbors. Candidate points closer to the prediction location than the most distant current neighbor can replace that neighbor. Green candidates decrease prediction error after replacement, whereas orange candidates increase it. All candidate additions remove the blue point at $(0.55, 0.24)$, yet there is no simple geometric rule separating the candidates that improve prediction from those that degrade it.

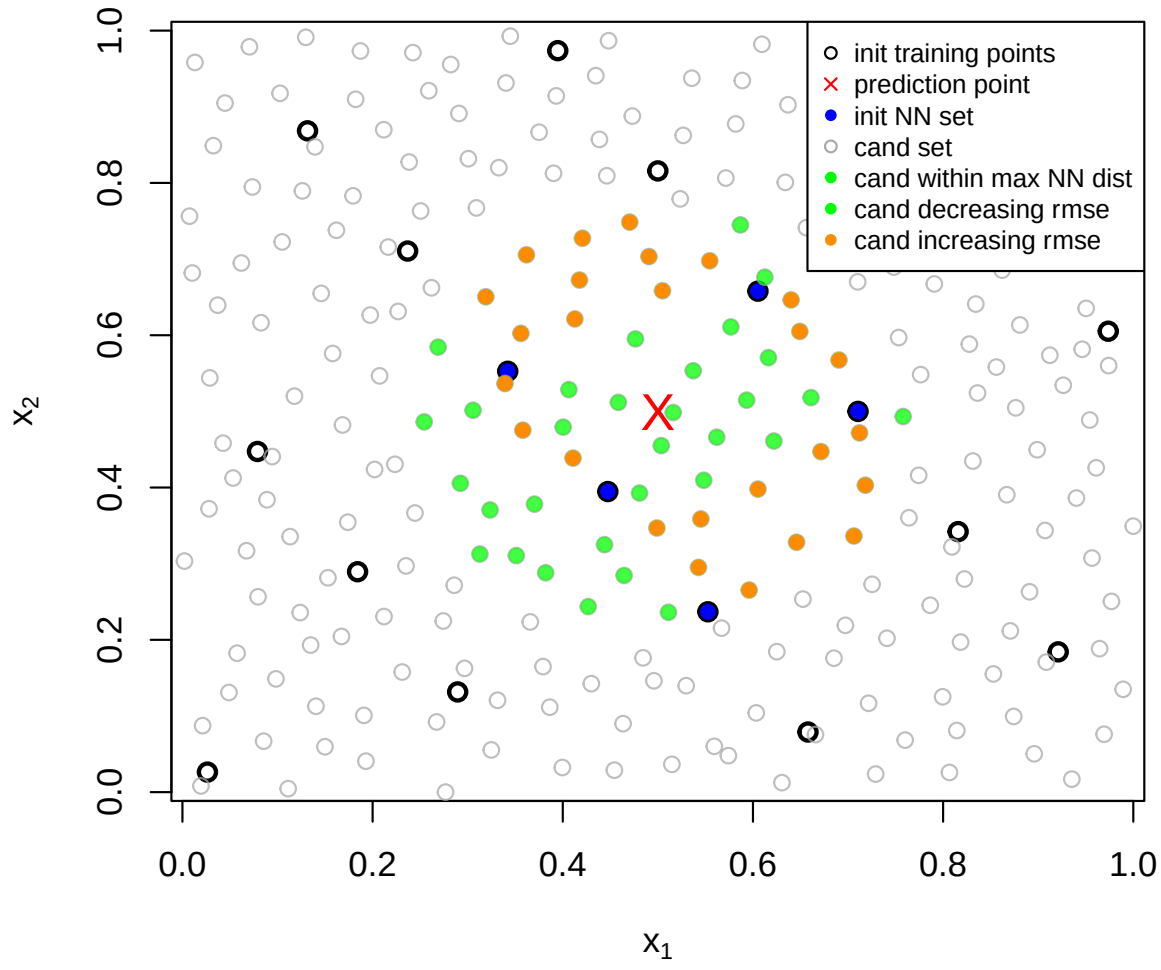


Figure SM1: Effect of replacing one point in a five-nearest-neighbor prediction set. The red × is the prediction location, black circles are the initial training points, and blue points form the current nearest-neighbor set. Candidate points inside the current maximum neighbor distance are colored green when the replacement decreases prediction error and orange when it increases prediction error.

SM3. Computational speedups from input scaling

A simulation study is used to illustrate the computational gains from input scaling and nearest-neighbor prediction relative to the default local approximate GP workflow. The manuscript identifies two benefits of input scaling. First, under an anisotropic correlation function, scaling makes ordinary nearest neighbors more closely aligned with the fitted GP correlation structure, avoiding repeated greedy neighborhood construction with the active-learning-Cohn (ALC) criterion. Second, global lengthscales are estimated once and reused rather than re-estimated at every prediction. This sacrifices some of laGP's nonstationary flexibility, but the timing experiments below indicate that the combined speedup can approach two orders of magnitude for problem sizes similar to the application.

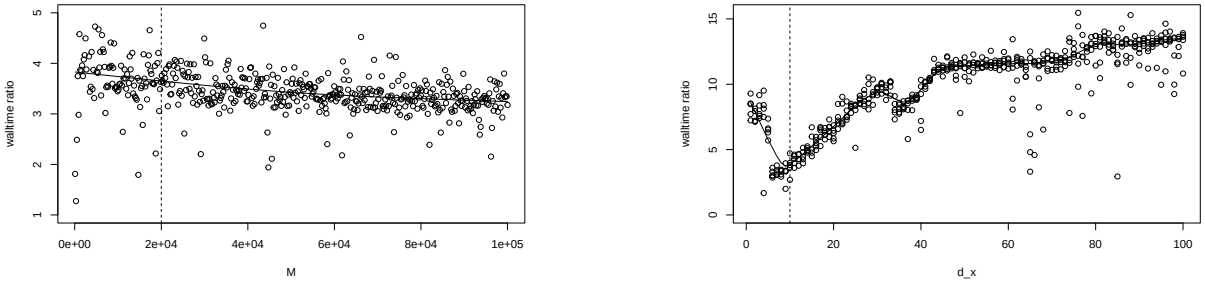
SM3.1. Neighborhood selection with laGP

The first experiment quantifies the additional cost of ALC neighborhood selection relative to simple nearest-neighbor selection. We report the wall-time ratio

$$\frac{t_{\mathrm{ALC}}}{t_{\mathrm{NN}}}.$$

We first fix $d_x = 10$ and generate 500 random datasets with ensemble sizes spanning $100 \leq M \leq 100000$. Input matrices and scalar outputs are generated independently and uniformly, and predictions are made sequentially at 10 locations using both ALC and nearest-neighbor neighborhood selection. Input scaling is not used, so local correlation parameters are estimated for each prediction.

A second set of 500 datasets uses $M = 20000$ and dimensions $1 \leq d_x \leq 100$, with five replicates at each dimension. Figure SM2 shows that ALC is generally three to five times slower than nearest-neighbor selection over a broad range of M when $d_x = 10$. The ratio is more variable as d_x changes and ranges from approximately 3 to 15. Vertical black lines indicate the application-scale setting $M = 20000$ and $d_x = 10$.



(a) Ratio as a function of ensemble size M.

(b) Ratio as a function of input dimension d_x.

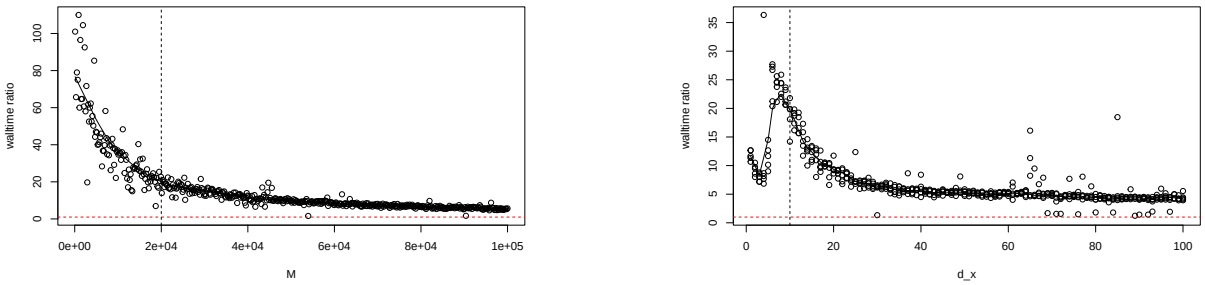
Figure SM2: Wall-time ratio $t_{\mathrm{ALC}}/t_{\mathrm{NN}}$ for ALC neighborhood selection relative to simple nearest-neighbor selection. For fixed d_x, the ratio decreases slightly with increasing M. For fixed M, it generally increases with d_x after approximately $d_x = 5$. Vertical black lines mark the approximate size of the application dataset.

SM3.2. Parameter estimation with laGP

Repeated local parameter estimation can be costly when predictions are made thousands of times during MCMC. The second experiment compares nearest-neighbor prediction with local parameter estimation against nearest-neighbor prediction using pre-estimated global lengthscales. We report the wall-time ratio

$$\frac{t_{\text{NN+local estimation}}}{t_{\text{NN+fixed lengthscales}}}.$$

The results are shown in Figure SM3. As either M or d_x becomes large, the ratio approaches one because nearest-neighbor search increasingly dominates parameter-estimation time. At the application-scale setting $M = 20000$ and $d_x = 10$, however, pre-estimating the lengthscales still provides an approximately $20\times$ speedup. Combining this with an approximately $5\times$ speedup from nearest-neighbor rather than ALC neighborhood selection gives an approximate two-order-of-magnitude speedup. The one-time cost of global lengthscale estimation is not negligible, but for the application considered here it is small relative to the cost of repeated prediction during Bayesian calibration.



(a) Ratio as a function of ensemble size M.

(b) Ratio as a function of input dimension d_x.

Figure SM3: Wall-time ratio for nearest-neighbor prediction with repeated local parameter estimation relative to nearest-neighbor prediction using fixed, pre-estimated global lengthscales. The horizontal red line indicates a ratio of one, and vertical black lines mark the approximate size of the application dataset.