*Proceedings of the 2020 Winter Simulation Conference*
*K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.*

# SMART LINEAR ALGEBRAIC OPERATIONS FOR EFFICIENT GAUSSIAN MARKOV IMPROVEMENT ALGORITHM

Xinru Li
Eunhye Song

Department of Industrial and Manufacturing Engineering
Pennsylvania State University
Leonhard Building
University Park, PA 16802, USA

## ABSTRACT

This paper studies computational improvement of the Gaussian Markov improvement algorithm (GMIA) whose underlying response surface model is a Gaussian Markov random field (GMRF). GMIA's computational bottleneck lies in the sampling decision, which requires factorizing and inverting a sparse, but large precision matrix of the GMRF at every iteration. We propose smart GMIA (sGMIA) that performs expensive linear algebraic operations intermittently, while recursively updating the vectors and matrices necessary to make sampling decisions for several iterations in between. The latter iterations are much cheaper than the former at the beginning, but their costs increase as the recursion continues and ultimately surpass the cost of the former. sGMIA adaptively decides how long to continue the recursion by minimizing the average per-iteration cost. We perform a floating-point operation analysis to demonstrate the computational benefit of sGMIA. Experiment results show that sGMIA enjoys computational efficiency while achieving the same search effectiveness as GMIA.

## 1 INTRODUCTION

Bayesian optimization (BO) has become a popular tool for global optimization of a black-box function whose analytical properties are seldom known. BO can be viewed as a simulation optimization methodology when the function can only be evaluated via simulation with stochastic error. BO draws inference from a stochastic process that models the response surface of the black-box function defined on the feasible solution space. Gaussian process (GP) has been adopted as the workhorse model for BO because of its convenience in updating the conditional distribution as well as its inferential power on the remaining uncertainty about the response surface.

Many BO algorithms are designed to solve an optimization problem defined on a continuous solution space. Thus, it is sensible to adopt a GP defined on a continuous input domain as its underlying model. In the simulation literature, continuous GP models have been adopted to design discrete optimization via simulation (DOvS) algorithms: see, for instance, Sun et al. (2014) and Xie et al. (2016). However, Salemi et al. (2019) point out that a typical choice of the covariance function for a continuous GP model may result in overconfidence about the response surface when applied to a DOvS problem.

As an alternative to the continuous GP-based algorithms, Salemi et al. (2019) design the Gaussian Markov Improvement Algorithm (GMIA) to solve a large-scale DOvS problem whose feasible solution space lies in the $d$-dimensional integer lattice. GMIA models the response surface as a realization of a Gaussian Markov random field (GMRF), a GP defined on a graph of nodes, where connectivity of nodes on the graph determines the correlation structure of the GMRF; a node's response is independent from the rest of the random field conditional on the neighboring nodes' (hence, Markovian). Salemi et al. (2019)

argue that this property is suitable for DOvS since each solution's performance can be inferred from its neighboring solutions' and less information is drawn from others.

Unlike a continuous GP whose conditional distribution update is made with respect to the covariance matrix, updating a GMRF involves its precision matrix, the inverse of the covariance matrix. Although updating a precision matrix of a GMRF is much cheaper than updating the covariance matrix of a continuous GP, elements of the covariance matrix are required to make a sampling decision for GMIA. Inverting a precision matrix can be costly when the feasible solution space is large. We show that such a naïve implementation can result in a higher per-iteration computational cost than a continuous GP. When simulation replications are cheap, high computational overhead may make GMIA less attractive than continuous GP-based algorithms. Motivated by this observation, the focus of this paper is to develop a procedure that significantly reduces the computational cost of GMIA.

Semelhago et al. (2017) recognize that only the diagonal and a single column of the covariance matrix are needed for the sampling decision of GMIA and dramatically reduce the cost by computing only the necessary elements of the covariance matrix. Semelhago et al. (2020) trade off search effectiveness to achieve further speed-up by restricting the sampling decisions within a promising subset of solutions for several rapid-search iterations. However, performance of their approach is quite sensitive to user-specified parameters such as the length of rapid-search iterations, which are difficult to optimize prior to running simulations.

We propose *smart* GMIA (sGMIA) that applies the Sherman-Morrison-Woodbury (SMW) formula to update only necessary elements of the inverse precision matrix. These elements are updated recursively at each iteration until it is no longer cheaper than factorizing and inverting the precision matrix. The length of the recursion is determined adaptively to minimize the per-iteration computational cost, which does not require any user inputs. sGMIA provides exact global inference at all feasible solutions at each iteration, thus it shows the same search effectiveness as the original GMIA with much improved computational efficiency.

The remainder of the paper is organized as follows. Section 2 provides a background on GP-based DOvS. Section 3 compares computational costs of updating continuous GP and GMRF models. In section 4, we propose two variations of sGMIA and analyze their computational efficiency. Section 5 introduces adaptive parameter selection for sGMIA. Section 6 evaluates empirical performance of sGMIA with benchmark algorithms, and conclusions are in Section 7.

## 2 GP-BASED DOVS

The DOvS problem of our interest can be written as

$$\min_{\mathbf{x}\in\mathscr{X}} y(\mathbf{x}) \triangleq \mathbb{E}[\mathrm{Y}(\mathbf{x})],$$

where $\mathscr{X}$ is a subset of $d$-dimensional integer lattice $\mathbb{Z}^d$ and $n \triangleq |\mathscr{X}|$ is the number of feasible solutions. We assume that $\mathscr{X}$ is a hyperbox and the distribution of $\mathrm{Y}(\mathbf{x})$ is unknown, but its expectation can be estimated via simulations. For any feasible solution $\mathbf{x}$, $\mathrm{Y}_j(\mathbf{x}) = y(\mathbf{x}) + \xi_j(\mathbf{x})$ can be observed for replications $j = 1, 2, \ldots$, where simulation error $\xi_j(\mathbf{x})$ is independent, identically distributed (i.i.d.) with zero mean and unknown variance $\sigma^2(\mathbf{x})$. To facilitate GP modeling, we further assume $\xi_j(\mathbf{x})$ is normally distributed.

For both continuous GP and GMRF, the unknown objective function values at all feasible solutions in $\mathscr{X}$ are modeled as multivariate Gaussian vector $\mathbb{Y}$. We define $\mathscr{X}_2 \subset \mathscr{X}$ to be the simulated solutions and $\mathscr{X}_1 \triangleq \mathscr{X} \setminus \mathscr{X}_2$. These two sets are updated at every iteration as more solutions are simulated. Let the sub-vector of $\mathbb{Y}$ corresponding to $\mathscr{X}_2$ be $\mathbb{Y}_2$ and $\mathbb{Y}_1 = \mathbb{Y} \setminus \mathbb{Y}_2$. Furthermore, let $n_1 = |\mathscr{X}_1|$ and $n_2 = |\mathscr{X}_2|$. For both continuous GP and GMRF, the prior joint distribution is

$$\mathbb{Y} = \begin{bmatrix} \mathbb{Y}_1 \\ \mathbb{Y}_2 \end{bmatrix} \sim \mathscr{N}\left( \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^\top & \Sigma_{22} \end{bmatrix} \right), \tag{1}$$

where $\mu_1$ and $\mu_2$ are mean vectors corresponding to $\mathbb{Y}_1$ and $\mathbb{Y}_2$, and $\Sigma$ is the covariance matrix divided into block matrices $\Sigma_{ij}$ for $1 \le i, j \le 2$ corresponding to $\mathscr{X}_1$ and $\mathscr{X}_2$. We define $\bar{\mathscr{Y}}_2$ as the vector of sample means at the solutions in $\mathscr{X}_2$. Since our problem has simulation error, $\bar{\mathscr{Y}}_2$ can be modeled as a realization of another GP, $\mathbb{Y}_2^\varepsilon = \mathbb{Y}_2 + \varepsilon$, where $\varepsilon \sim \mathcal{N}(\mathbf{0}_{n_2 \times 1}, \Sigma_\varepsilon)$. Here $\Sigma_\varepsilon$ is a diagonal matrix if all solutions are simulated independently. We use the sample variance of the sample mean at each design point as a plug-in estimate of the corresponding diagonal element of $\Sigma_\varepsilon$. Conditional on $\mathbb{Y}_2 = \bar{\mathscr{Y}}_2$, (1) is updated. The computation required for the conditional distribution update is different for a continuous GP and a GMRF, which we discuss in detail in Section 3.

Frazier (2018) reviews that there are three major sampling criteria for BO: expected improvement (EI), knowledge-gradient (KG) and entropy search (ES). First proposed by Jones et al. (1998), EI measures the expected improvement in the response with respect to the current best if a solution other than the current best is selected. Because EI is defined for a deterministic computer experiment code, it does not reflect the uncertainty in the response at the current best. Accounting for such uncertainty, Salemi et al. (2019) modify EI to define the complete expected improvement (CEI). In this paper, we focus on computational cost comparison when CEI is adopted as the sampling criterion.

We define the current best, $\tilde{\mathbf{x}}$, as the solution with the smallest sample mean. For $\mathbf{x} \in \mathscr{X}$, let $\mathbf{M}(\mathbf{x})$, $\mathbf{V}(\mathbf{x})$ and $\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x})$ be conditional mean of $\mathbf{x}$, conditional variance of $\mathbf{x}$, and conditional covariance between $\mathbf{x}$ and $\tilde{\mathbf{x}}$, respectively. Then, the conditional variance of the difference $\mathbb{Y}(\tilde{\mathbf{x}}) - \mathbb{Y}(\mathbf{x})$ is $\mathbf{V}(\tilde{\mathbf{x}}, \mathbf{x}) = \mathbf{V}(\tilde{\mathbf{x}}) + \mathbf{V}(\mathbf{x}) - 2\mathbf{C}(\tilde{\mathbf{x}}, \mathbf{x})$. Salemi et al. (2019) derive the CEI of $\mathbf{x}$ is

$$\begin{aligned} \mathrm{CEI}(\mathbf{x}) &= \mathbb{E}[\max(\mathbb{Y}(\tilde{\mathbf{x}}) - \mathbb{Y}(\mathbf{x}), 0)] \\ &= (\mathbf{M}(\tilde{\mathbf{x}}) - \mathbf{M}(\mathbf{x}))\Phi\left(\frac{\mathbf{M}(\tilde{\mathbf{x}}) - \mathbf{M}(\mathbf{x})}{\sqrt{\mathbf{V}(\tilde{\mathbf{x}}, \mathbf{x})}}\right) + \sqrt{\mathbf{V}(\tilde{\mathbf{x}}, \mathbf{x})}\,\phi\left(\frac{\mathbf{M}(\tilde{\mathbf{x}}) - \mathbf{M}(\mathbf{x})}{\sqrt{\mathbf{V}(\tilde{\mathbf{x}}, \mathbf{x})}}\right), \end{aligned} \tag{2}$$

where $\phi$ and $\Phi$ are the probability density and cumulative distribution function, respectively, of a standard normal random variable. Equation (2) shows that it is sufficient to update the conditional mean and the diagonal and the column corresponding to $\tilde{\mathbf{x}}$ of the covariance matrix to compute CEIs at all solutions in $\mathscr{X}$. GMIA simulates $\tilde{\mathbf{x}}$ as well as the CEI-maximizing solution, $\mathbf{x}_{\mathrm{CEI}}$, at each iteration.

## 3 COMPUTATIONAL COMPARISON OF CONTINUOUS GP AND GMRF INFERENCE

When the cost of simulation replication is mild to moderate, the cost of linear algebraic operations involved in inference can dominate the overall computational cost of the algorithm. In the following sections, we review continuous GP and GMRF models and analyze their computational costs for CEI computation.

### 3.1 CONTINUOUS GP UPDATE AND INFERENCE

The prior covariance matrix, $\Sigma$, of a continuous GP is characterized by its covariance kernel $k(\mathbf{x}, \mathbf{x}') \triangleq \mathbb{E}[(\mathbb{Y}(\mathbf{x}) - \mu(\mathbf{x}))(\mathbb{Y}(\mathbf{x}') - \mu(\mathbf{x}'))]$, where $\Sigma$ is the Gram matrix whose $(i, j)$th element is $k(\mathbf{x}_i, \mathbf{x}_j)$ (Rasmussen and Williams 2006). Typically, $k(\mathbf{x}, \mathbf{x}') > 0$, which makes $\Sigma$ a dense matrix with positive entries. Let $\mathbf{K}(\mathscr{A}, \mathscr{B})$ be the Gram matrix whose entries are $\mathrm{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for $\mathbf{x}_i \in \mathscr{A}$ and $\mathbf{x}_j \in \mathscr{B}$. Thus, $\Sigma_{11}$ in (1) is $\mathbf{K}(\mathscr{X}_1, \mathscr{X}_1)$. Similarly, $\Sigma_{12} = \mathbf{K}(\mathscr{X}_1, \mathscr{X}_2)$ and $\Sigma_{22} = \mathbf{K}(\mathscr{X}_2, \mathscr{X}_2)$.

The conditional distribution of $\mathbb{Y}$ given $\mathbb{Y}_2^\varepsilon = \bar{\mathscr{Y}}_2$ is $\mathcal{N}(\bar{\mu}, \bar{\Sigma})$, where

$$\begin{aligned} \bar{\mu} &= \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \Sigma_{12} \\ \Sigma_{22} \end{bmatrix} (\Sigma_{22} + \Sigma_\varepsilon)^{-1} (\bar{\mathscr{Y}}_2 - \mu_2), \\ \bar{\Sigma} &= \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^\top & \Sigma_{22} \end{bmatrix} - \begin{bmatrix} \Sigma_{12} \\ \Sigma_{22} \end{bmatrix} (\Sigma_{22} + \Sigma_\varepsilon)^{-1} \begin{bmatrix} \Sigma_{12} & \Sigma_{22} \end{bmatrix}. \end{aligned} \tag{3}$$

Notice that both conditional mean and covariance matrix require computing $(\Sigma_{22} + \Sigma_\varepsilon)^{-1}[\Sigma_{12}\ \Sigma_{22}]$. An efficient way to compute $\mathbf{C} = \mathbf{A}^{-1}\mathbf{B}$, where $\mathbf{A}$ is a symmetric positive definite $m_1 \times m_1$ matrix, and $\mathbf{B}$ and

$\mathbf{C}$ are $m_1 \times m_2$ matrices, is using backsolve. The steps for backsolve consist of 1) computing the Cholesky factorization $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$ where $\mathbf{L}$ is a lower triangular matrix, which costs $\frac{1}{3}m_1^3$ floating point operations (flops) for dense $\mathbf{A}$; 2) solving $\mathbf{L}\mathbf{C}_1 = \mathbf{B}$ using forward substitution, which costs $m_2 m_1^2$ flops; 3) solving $\mathbf{L}^\top \mathbf{C} = \mathbf{C}_1$ using backward substitution, which again costs $m_2 m_1^2$ flops. Therefore, the overall cost of backsolve is $\frac{1}{3}m_1^3 + 2m_2 m_1^2$. Back to our problem, $(\Sigma_{22} + \Sigma_\varepsilon)$ is a dense positive definite $n_2 \times n_2$ matrix, and $[\Sigma_{12}\ \Sigma_{22}]$ is an $n_2 \times n$ matrix, so the computational cost of $(\Sigma_{22} + \Sigma_\varepsilon)^{-1}[\Sigma_{12}\ \Sigma_{22}]$ is $2nn_2^2 + \frac{1}{3}n_2^3$. Then, $\begin{bmatrix} \Sigma_{12} \\ \Sigma_{22} \end{bmatrix} (\Sigma_{22} + \Sigma_\varepsilon)^{-1}(\bar{\mathscr{Y}}_2 - \mu_2)$ is a product of an $n \times n_2$ matrix and an $n_2 \times 1$ vector, which costs $2nn_2 - n$ flops. Thus, the overall cost of computing the conditional mean is $2nn_2^2 + 2nn_2 + \frac{1}{3}n_2^3$ flops.

For CEI computation, recall that only the diagonal and a single column of $\bar{\Sigma}$ are needed. From (3), note that updating the $(i,j)$th element of $\bar{\Sigma}$ requires the inner product of the $i$th row of $\begin{bmatrix} \Sigma_{12} \\ \Sigma_{22} \end{bmatrix}$ and the $j$th column of $(\Sigma_{22} + \Sigma_\varepsilon)^{-1}[\Sigma_{12}\ \Sigma_{22}]$. Given the latter matrix, this costs $2n_2 - 1$ flops. Thus, the cost of computing conditional variance and covariance is $4nn_2$ flops. Combined with the computational cost of the conditional mean, the total cost of updating a continuous GP is $\mathscr{O}(nn_2^2)$, which increases as the number of simulated solutions, $n_2$, increases.

Frazier et al. (2011) propose computing the conditional mean $\mu_i$ and covariance matrix $\Sigma_i$ recursively by implementing the SMW formula. Suppose at the $(i-1)$th iteration, we find optimal solution $\mathbf{x}_1^{i-1}$ and max CEI solution $\mathbf{x}_2^{i-1}$ to simulate and $\bar{\mathscr{Y}}_{i-1}$ is a vector of the outputs for them. Let $\mathbf{U}_{i-1} = [\mathbf{e}_{\mathbf{x}_1^{i-1}}\ \mathbf{e}_{\mathbf{x}_2^{i-1}}]$ where $\mathbf{e}_{\mathbf{x}}$ is the standard basis vector consisting of 0s and a 1 at the row corresponding to $\mathbf{x}$. Then,

$$\begin{aligned} \Sigma_i &= \Sigma_{i-1} - \Sigma_{i-1}\mathbf{U}_{i-1}^\top (\mathbf{U}_{i-1}\Sigma_{i-1}\mathbf{U}_{i-1}^\top + \Sigma_\varepsilon^i)^{-1}\mathbf{U}_{i-1}\Sigma_{i-1}, \\ \mu_i &= \left[\mathbf{I}_{n\times n} - \Sigma_{i-1}\mathbf{U}_{i-1}^\top (\mathbf{U}_{i-1}\Sigma_{i-1}\mathbf{U}_{i-1}^\top + \Sigma_\varepsilon^i)^{-1}\mathbf{U}_{i-1}\right]\mu_{i-1} + \Sigma_i \mathbf{U}_{i-1}^\top (\Sigma_\varepsilon^i)^{-1}\bar{\mathscr{Y}}_{i-1}, \end{aligned} \tag{4}$$

where $\mathbf{I}_{n\times n}$ is the $n \times n$ identity matrix. Note that $\mathbf{U}_{i-1}\Sigma_{i-1}\mathbf{U}_{i-1}^\top + \Sigma_\varepsilon^i$ is a $2 \times 2$ matrix, so the cost of computing its inverse is negligible. From a similar analysis as above, one can show that the total computation costs of updating $\Sigma_i$ and $\mu_i$ are $4n^2 + 6n$ and $2n^2 + 3n$ flops, respectively. Notice that the total computation cost is $\mathscr{O}(n^2)$. Therefore, as iterations continue and more solutions are simulated, SMW update scheme in (4) eventually becomes cheaper than (3), which costs $\mathscr{O}(nn_2^2)$. Updating only the diagonal and a single column of $\Sigma_i$ is less straightforward for (4) because in the $(i+1)$th iteration, computing $\Sigma_i \mathbf{U}_i^\top$ requires the columns of $\Sigma_i$ corresponding to $\mathbf{x}_1^i$ and $\mathbf{x}_2^i$ and these solutions may change every iteration.

### 3.2 GMRF UPDATE AND INFERENCE

A GMRF is a special case of GP, defined on an undirected labelled graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is the set of nodes and $\mathscr{E}$ is the set of edges connecting the nodes (Rue and Held 2005). The covariance matrix of the GMRF is $\Sigma = \mathbf{Q}^{-1}$, where $\mathbf{Q}$ is the precision matrix. The entries of precision matrix reflects the neighborhood structure of $\mathscr{G}$. Let $\mathbb{Y}_{\mathscr{V}\setminus S}$ be the sub-vector of $\mathbb{Y}$ except for the nodes in set $S$. The diagonals of the precision matrix are conditional precision $\mathrm{Q}_{ii} = \mathrm{Prec}(\mathbb{Y}_i | \mathbb{Y}_{\mathscr{V}\setminus\{i\}}) = \frac{1}{\mathbf{V}(\mathbb{Y}_i | \mathbb{Y}_{\mathscr{V}\setminus\{i\}})}$. The off-diagonals are proportional to the conditional correlation

$$\mathrm{Q}_{ij} = -\mathrm{Corr}(\mathbb{Y}_i, \mathbb{Y}_j | \mathbb{Y}_{\mathscr{V}\setminus\{i,j\}})\sqrt{\mathrm{Q}_{ii}\mathrm{Q}_{jj}}, \quad i \neq j. \tag{5}$$

The Markov property of GMRF gives $\mathbb{Y}_i \perp \mathbb{Y}_{\mathscr{V}\setminus\{\mathrm{N}(i),i\}} | \mathbb{Y}_{\mathrm{N}(i)}$ for every $i \in \mathscr{V}$, where $\mathrm{N}(i)$ are the neighbors of Node $i$. In words, this means given the responses at all neighboring nodes of Node $i$, the response at Node $i$ is independent from the rest of the field's. Combining this property with (5), $\mathrm{Q}_{ij} = 0$, if $j \notin \mathrm{N}(i)$. We adopt the neighborhood structure chosen by Salemi et al. (2019): $\mathrm{N}(\mathbf{x}) = \{\mathbf{x}' \in \mathscr{X} : \| \mathbf{x} - \mathbf{x}' \|_2 = 1\}$. This choice of $\mathrm{N}(\cdot)$ makes $\mathbf{Q}$ very sparse with fill-in rate no more than $\frac{2d+1}{n}$. With this neighborhood

structure, we parameterize $\mathbf{Q}$ so that its $(i,j)$th entry is

$$\mathrm{Q}_{ij} \triangleq \begin{cases} \theta_0, & \text{if } i = j, \\ -\theta_0\theta_k, & \text{if } |\mathbf{x}_i - \mathbf{x}_j| = \mathbf{e}_k, \\ 0, & \text{otherwise,} \end{cases}$$

for $\mathbf{x}_i, \mathbf{x}_j \in \mathscr{X}$, where $\mathbf{e}_k$ is the $k$th standard basis vector and $|\cdot|$ is the element-wise absolute value operation. The conditional precision of any solution must be positive by definition, so $\theta_0 = \mathrm{Q}_{ii} > 0$. We typically assume the prior correlation among two solutions to be positive, which gives $\theta_j > 0$ for all $j$. Moreover, $\sum_{j=1}^{d} \theta_j < 0.5$ makes $\mathbf{Q}$ positive definite. Although $\mathbf{Q}$ is sparse, its inverse $\Sigma$ is dense.

The GMIA models the response surface of $\mathscr{X}$ as a GMRF

$$\begin{bmatrix} \mathbb{Y}_1 \\ \mathbb{Y}_2 \end{bmatrix} = \mathscr{N}\left( \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{12}^\top & \mathbf{Q}_{22} \end{bmatrix}^{-1} \right),$$

where $\mathbf{Q}_{ij}, 1 \le i,j \le 2$, are block matrices corresponding to $\mathscr{X}_1$ and $\mathscr{X}_2$. Similar to the continuous GP model, $\mathbb{Y}_2^\varepsilon = \mathbb{Y}_2 + \varepsilon$, where $\varepsilon \sim \mathscr{N}(\mathbf{0}_{n_2\times 1}, \mathbf{Q}_\varepsilon^{-1})$ and $\mathbf{Q}_\varepsilon$ is the precision matrix of $\varepsilon$. We choose to simulate all solutions independently, so $\mathbf{Q}_\varepsilon$ is diagonal. The element of $\mathbf{Q}_\varepsilon$ corresponding to $\mathbf{x} \in \mathscr{X}_2$ is approximated by $1/\mathrm{V}(\bar{\mathscr{Y}}_2(\mathbf{x}))$, where $\mathrm{V}(\bar{\mathscr{Y}}_2(\mathbf{x}))$ is the sample variance of the average simulation output at $\mathbf{x}$ as a plug-in estimate. Salemi et al. (2019) prove that the conditional distribution of $\mathbb{Y}$ given $\mathbb{Y}_2^\varepsilon = \bar{\mathscr{Y}}_2$ is

$$\mathscr{N}\left( \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \bar{\mathbf{Q}}^{-1} \begin{bmatrix} \mathbf{0}_{n_1\times 1} \\ \mathbf{Q}_\varepsilon(\bar{\mathscr{Y}}_2 - \mu_2) \end{bmatrix}, \bar{\mathbf{Q}}^{-1} \right), \tag{6}$$

where

$$\bar{\mathbf{Q}} \triangleq \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{12}^\top & \mathbf{Q}_{22} + \mathbf{Q}_\varepsilon \end{bmatrix} \tag{7}$$

is the conditional precision matrix.

To update the conditional mean, $\bar{\mathbf{Q}}^{-1} \begin{bmatrix} \mathbf{0}_{n_1\times 1} \\ \mathbf{Q}_\varepsilon(\bar{\mathscr{Y}}_2 - \mu_2) \end{bmatrix}$ is needed. Because the sparsity pattern of $\bar{\mathbf{Q}}$ is exactly the same as that of $\mathbf{Q}$ from (7), the Cholesky factorization of $\bar{\mathbf{Q}}$ is much cheaper than $\frac{1}{3}n^3$. Computation software such as Matlab first permutes a sparse matrix to make the Cholesky factorization the most efficient, therefore, analytically obtaining the exact flop counts is difficult. From a Matlab time study, we observed that Cholesky factorization costs approximately $\mathscr{O}(n^{1.6})$ for a two-dimensional problem, where the order is obtained from a log-log plot of the computation time vs. $n$. For higher dimensions, the cost is higher as $\bar{\mathbf{Q}}$ is denser. From the same time study, a single backsolve cost was estimated to be $\mathscr{O}(n^{1.4})$. Consequently, computing the full inverse, $\bar{\mathbf{Q}}^{-1}$, using $n$ backsolves costs $\mathscr{O}(n^{2.4})$, which dominates all other costs.

This analysis shows that a naïve update of the GMRF conditional distribution results in higher per-iteration computational cost than the conitnuous-GP. Computing only the diagonal and a single column of $\bar{\mathbf{Q}}^{-1}$ significantly reduces the computational cost as shown in the following section.

## 4 SMART LINEAR ALGEBRAIC OPERATIONS FOR GMIA

Semelhago et al. (2017) propose an efficient way to compute the diagonal of $\bar{\mathbf{Q}}^{-1}$ without computing its full inverse. They first perform the LDL factorization, $\bar{\mathbf{Q}} = \mathbf{L}\mathbf{D}\mathbf{L}^\top$, where $\mathbf{L}$ is a sparse lower triangular matrix with ones on its diagonal, and $\mathbf{D}$ is a diagonal matrix. They exploit the following elementwise identity (Takahashi et al. 1973); $\Sigma_{ii} = \mathbf{D}_{ii}^{-1} - \sum_{k>i} \mathbf{L}_{ki}\Sigma_{ki}$ and $\Sigma_{ij} = \sum_{k>i} \mathbf{L}_{ki}\Sigma_{kj}$, $\forall i < j$. Thus, only a few elements of $\Sigma$ need to be computed to obtain the diagonal of $\bar{\mathbf{Q}}^{-1}$ when $\mathbf{L}$ is sparse such that most of $\mathbf{L}_{ki}$

is 0. A linear solver, PARDISO, implements this method; for algorithmic details of PARDISO see Petra et al. (2014a) and Petra et al. (2014b). Using PARDISO significantly reduces the cost of computing the diagonal of $\bar{\mathbf{Q}}^{-1}$, however, it still requires LDL factorization of $\bar{\mathbf{Q}}$ at every iteration.

Semelhago et al. (2020) propose rapid Gaussian Markov Improvement Algorithm (rGMIA), which factorizes $\bar{\mathbf{Q}}$ and extracts the diagonal of $\bar{\mathbf{Q}}^{-1}$ periodically. rGMIA separates all feasible solutions into two disjoint sets, $\mathscr{S}$ and $\mathscr{F}$, where $\mathscr{S}$ contains more promising solutions and is much smaller than $\mathscr{F}$. There are two search stages in rGMIA: global search computes the CEIs of solutions in both $\mathscr{S}$ and $\mathscr{F}$; rapid search only computes the CEIs for solutions in $\mathscr{S}$. One global-search iteration is followed by $p-1$ rapid-search iterations. Because $|\mathscr{S}| \ll |\mathscr{F}|$, the per-iteration computation cost is greatly reduced compared to Semelhago et al. (2017). One drawback of rGMIA is that the size of $\mathscr{S}$ is difficult to optimize; if too small, it hinders search effectiveness and if too large, the computational benefit may not be fully exploited. They recommend $|\mathscr{S}| = p$ based on experiment results, but this is heuristic. The optimal choice of $p$ is another open question. rGMIA also utilizes PARDISO, although it can be implemented without PARDISO.

We propose sGMIA that avoids factorizing $\bar{\mathbf{Q}}$ at every iteration while computing the CEI at all solutions at each iteration. Moreover, unlike rGMIA, sGMIA does not require user-specified parameters and adaptively selects the algorithm parameter to achieve maximal computational efficiency. Although PARDISO provides free license for academic users, it requires compiling source files with C++ and Fortran compilers, which may be a hurdle for less computationally-savvy users. Thus, we present two variations of sGMIA, with and without PARDISO.

Similar to rGMIA, there are two types of sGMIA iterations: *full* and *SMW* iterations. Table 1 summarizes the linear algebraic operations performed at each full iteration of sGMIA with PARDISO. Let $\tilde{\mathbf{x}}^t$, $\bar{\mathbf{Q}}^t$, $\mathbf{Q}_\varepsilon^t$ and $\bar{\mu}^t$ be the current best solution at the $t$th iteration, the GMRF precision matrix, the simulation error precision matrix of the solutions in $\mathscr{X}_2$, and the conditional mean at the $t$th iteration, respectively. Using PARDISO, LDL factorization of $\bar{\mathbf{Q}}^t$ is first performed, then the diagonal of $(\bar{\mathbf{Q}}^t)^{-1}$ is computed. Two backsolve operations using LDL factors are performed to compute the conditional covariance vector corresponding to $\tilde{\mathbf{x}}^t$, and the conditional mean. Table 1 presents the computational cost of each step. We use $C_f, C_d$, and $C_b$ to denote the costs of LDL factorization, computing the diagonal of the inverse, and a single-vector backsolve operation, respectively, by PARDISO. As mentioned earlier, exact flops of these operations are difficult to analyze. From time studies in Semelhago et al. (2020), $C_f, C_d$ and $C_b$ are estimated to be $\mathscr{O}(n^{1.4}), \mathscr{O}(n^{1.3})$ and $\mathscr{O}(n^1)$, respectively, for a two-dimensional problem. The full iteration ends with computing the CEI at all solutions and simulating the maximum CEI solution, $\mathbf{x}_{\text{CEI}}^t$, as well as $\tilde{\mathbf{x}}^t$.

In between two full iterations, sGMIA performs several SMW iterations while tracking the cumulative changes to $\bar{\mathbf{Q}}$ since the last full iteration. Suppose sGMIA performs a full iteration at the $t$th iteration. In the next iteration, $\bar{\mathbf{Q}}^t$ is updated to $\bar{\mathbf{Q}}^{t+1} = \bar{\mathbf{Q}}^t + \Delta_{\mathbf{x}_{\text{CEI}}^t} \mathbf{e}_{\mathbf{x}_{\text{CEI}}^t} \mathbf{e}_{\mathbf{x}_{\text{CEI}}^t}^\top + \Delta_{\tilde{\mathbf{x}}^t} \mathbf{e}_{\tilde{\mathbf{x}}^t} \mathbf{e}_{\tilde{\mathbf{x}}^t}^\top$, where $\Delta_{\mathbf{x}} = 1/\mathrm{V}(\bar{\mathscr{Y}}_2^{t+1}(\mathbf{x})) - 1/\mathrm{V}(\bar{\mathscr{Y}}_2^t(\mathbf{x}))$, if $\mathbf{x}$ is simulated in the first $t$ iterations and $\Delta_{\mathbf{x}} = 1/\mathrm{V}(\bar{\mathscr{Y}}_2^{t+1}(\mathbf{x}))$, otherwise. Note that $\Delta_{\mathbf{x}_{\text{CEI}}^t}$ and $\Delta_{\tilde{\mathbf{x}}^t}$ reflect the changes in the diagonal elements of $\mathbf{Q}_\varepsilon$ corresponding to $\mathbf{x}_{\text{CEI}}^t$ and $\tilde{\mathbf{x}}^t$, respectively. Therefore, $\bar{\mathbf{Q}}^{t+1}$ has exactly two elements different from $\bar{\mathbf{Q}}^t$, which motivates the use of the SMW formula.

Table 1: The flop analysis for a full iteration of sGMIA with PARDISO

| | |
|---|---|
| I. **LDL factorization** $\quad \bar{\mathbf{Q}}^t = \mathbf{L}\mathbf{D}\mathbf{L}^\top$ | $C_f$ |
| II. **Conditional variance** $\quad \mathrm{diag}((\bar{\mathbf{Q}}^t)^{-1})$ | $C_d$ |
| III. **Conditional covariance** $\quad \bar{\mathbf{Q}}^t \backslash \mathbf{e}_{\tilde{\mathbf{x}}}$ | $C_b$ |
| IV. **Conditional mean** $\quad \bar{\mu}^t = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \bar{\mathbf{Q}}^t \backslash \begin{bmatrix} \mathbf{0} \\ \mathbf{Q}_\varepsilon^t(\bar{\mathscr{Y}}_2 - \mu_2) \end{bmatrix}$ | $C_b + n$ |
| **Total computation cost:** $C_f + C_d + 2C_b + n$ flops | |

Let $\mathbf{U}_1 = [\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}}\ \mathbf{e}_{\tilde{\mathbf{x}}^t}]$ and $\mathbf{V}_1 = [\Delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}}\ \Delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}]$. Then, $\bar{\mathbf{Q}}^{t+1} = \bar{\mathbf{Q}}^t + \mathbf{U}_1\mathbf{V}_1^\top$ and

$$(\bar{\mathbf{Q}}^{t+1})^{-1} = (\bar{\mathbf{Q}}^t)^{-1} - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\left(\mathbf{I}_{2\times 2} + \mathbf{V}_1^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\right)^{-1}\mathbf{V}_1^\top(\bar{\mathbf{Q}}^t)^{-1}. \tag{8}$$

Recall that our goal is to maximize efficiency by computing only necessary elements. To compute the diagonal of $(\bar{\mathbf{Q}}^{t+1})^{-1}$, it suffices to compute $\text{diag}(\bar{\mathbf{Q}}^{t+1})^{-1} = \text{diag}(\bar{\mathbf{Q}}^t)^{-1} - \text{diag}\left((\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\mathbf{A}_1\right)$, where $\mathbf{A}_1 \triangleq \left(\mathbf{I}_{2\times 2} + \mathbf{V}_1^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\right)^{-1}\mathbf{V}_1^\top(\bar{\mathbf{Q}}^t)^{-1}$ and $\text{diag}(\bar{\mathbf{Q}}^t)^{-1}$ is already computed in the $t$th iteration. Note that $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1$ are two columns of $(\bar{\mathbf{Q}}^t)^{-1}$, which can be computed from backsolve operations by PARDISO. Exploiting that $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\mathbf{A}_1$ is a product of $n\times 2$ matrix $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1$ and $2\times n$ matrix $\mathbf{A}_1$, the $i$th diagonal element of $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\mathbf{A}_1$ can be computed from the inner product of the $i$th row of $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1$ and the $i$th column of $\mathbf{A}_1$. For the conditional covariance vector, $(\bar{\mathbf{Q}}^{t+1})^{-1}\mathbf{e}_{\tilde{\mathbf{x}}^{t+1}} = (\bar{\mathbf{Q}}^t)^{-1}\mathbf{e}_{\tilde{\mathbf{x}}^{t+1}} - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\mathbf{A}_1\mathbf{e}_{\tilde{\mathbf{x}}^{t+1}}$, where both terms can be computed from the backsolve operations with PARDISO.

Next, we discuss the conditional mean update. With a slight abuse of notation, let $\bar{\mathscr{Y}}^t$ be an $n\times 1$ vector whose elements corresponding to the simulated solutions are equal to their sample means, and the rest are equal to the corresponding elements of $\mu$. Let $\delta_{\mathbf{x}}$ be $\delta_{\mathbf{x}} = \bar{\mathscr{Y}}^{t+1}(\mathbf{x}) - \bar{\mathscr{Y}}^t(\mathbf{x})$, where $\bar{\mathscr{Y}}^t(\mathbf{x})$ is the element of $\bar{\mathscr{Y}}^t$ corresponding to $\mathbf{x}$. Then, the conditional mean update in (6) can be written as

$$\bar{\mu}^{t+1} = \mu + (\bar{\mathbf{Q}}^{t+1})^{-1}\left(\mathbf{Q}^t_\varepsilon + \Delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}^\top_{\mathbf{x}^t_{\text{CEI}}} + \Delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}\mathbf{e}^\top_{\tilde{\mathbf{x}}^t}\right)\left(\bar{\mathscr{Y}}^t - \mu + \delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}} + \delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}\right). \tag{9}$$

Let $\mathbf{b}_1$ satisfy $\left(\mathbf{Q}^t_\varepsilon + \Delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}^\top_{\mathbf{x}^t_{\text{CEI}}} + \Delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}\mathbf{e}^\top_{\tilde{\mathbf{x}}^t}\right)\left(\bar{\mathscr{Y}}^t - \mu + \delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}} + \delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}\right) = \mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t - \mu) + \mathbf{b}_1$. From simple algebra, one can show

$$\begin{aligned}\mathbf{b}_1 = {} & \left(\Delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}^\top_{\mathbf{x}^t_{\text{CEI}}} + \Delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}\mathbf{e}^\top_{\tilde{\mathbf{x}}^t}\right)(\bar{\mathscr{Y}}^t - \mu) + \mathbf{Q}^t_\varepsilon\left(\delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}} + \delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}\right) \\ & + \left(\Delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}^\top_{\mathbf{x}^t_{\text{CEI}}} + \Delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}\mathbf{e}^\top_{\tilde{\mathbf{x}}^t}\right)\left(\delta_{\mathbf{x}^t_{\text{CEI}}}\mathbf{e}_{\mathbf{x}^t_{\text{CEI}}} + \delta_{\tilde{\mathbf{x}}^t}\mathbf{e}_{\tilde{\mathbf{x}}^t}\right).\end{aligned} \tag{10}$$

Because $\mathbf{b}_1$ is a vector with only two nonzero entries corresponding to $\mathbf{x}^t_{\text{CEI}}$ and $\tilde{\mathbf{x}}^t$, it costs only a few flops to compute. Thus, the computational bottleneck of (9) is $(\bar{\mathbf{Q}}^{t+1})^{-1}\left(\mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t - \mu) + \mathbf{b}_1\right)$. Substituting $(\bar{\mathbf{Q}}^{t+1})^{-1}$ with $(\bar{\mathbf{Q}}^t)^{-1} - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\mathbf{A}_1$,

$$\begin{aligned}&\left((\bar{\mathbf{Q}}^t)^{-1} - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\mathbf{A}_1\right)\left(\mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t - \mu) + \mathbf{b}_1\right) \\ &= (\bar{\mathbf{Q}}^t)^{-1}\mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t - \mu) + (\bar{\mathbf{Q}}^t)^{-1}\mathbf{b}_1 - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1\mathbf{A}_1\left(\mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t - \mu) + \mathbf{b}_1\right),\end{aligned}$$

where the first term is already computed in the last iteration (See Part IV in Table 1) and the second term is a linear combination of two columns of $(\bar{\mathbf{Q}}^t)^{-1}$ corresponding to $\tilde{\mathbf{x}}^t$ and $\mathbf{x}^t_{\text{CEI}}$. The last term is a product of $n\times 2$ matrix $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_1$ and $2\times 1$ vector $\mathbf{A}_1\left(\mathbf{Q}^t_\varepsilon(\bar{\mathscr{Y}}^t - \mu) + \mathbf{b}_1\right)$, where the former is already computed for the conditional variance update.

We extend the same idea to the $(t+i)$th iteration for $i > 1$. Suppose there are $m$ distinct solutions, $\mathbf{x}_1, \ldots, \mathbf{x}_m$, simulated between the $t$th and the $(t+i)$th iterations. Note that $m \le 2i$ because sGMIA samples the current best and the maximum CEI solution at each iteration. Define $\mathbf{U}_i = [\mathbf{e}_{\mathbf{x}_1}\ \ldots\ \mathbf{e}_{\mathbf{x}_m}]$ and $\mathbf{V}_i = [\Delta_{\mathbf{x}_1}\mathbf{e}_{\mathbf{x}_1}\ \ldots\ \Delta_{\mathbf{x}_m}\mathbf{e}_{\mathbf{x}_m}]$, where $\Delta$s record the changes in $\mathbf{Q}_\varepsilon$ from $t$th iteration. Then $\mathbf{U}_i\mathbf{V}_i^\top = \bar{\mathbf{Q}}^{t+i} - \bar{\mathbf{Q}}^t$ is a matrix with $m$ nonzero diagonal elements and 0s elsewhere. Similar to (8), we have

$$(\bar{\mathbf{Q}}^{t+i})^{-1} = (\bar{\mathbf{Q}}^t)^{-1} - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\left(\mathbf{I}_{m\times m} + \mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\right)^{-1}\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1},$$

which is then used to compute the diagonal of $(\bar{\mathbf{Q}}^{t+i})^{-1}$ and $(\bar{\mathbf{Q}}^{t+i})^{-1}\mathbf{e}_{\tilde{\mathbf{x}}^{t+1}}$. The conditional mean is updated as

$$\bar{\mu}^{t+i} = \mu + (\bar{\mathbf{Q}}^{t+i})^{-1}\left(\mathbf{Q}^t_\varepsilon + \sum_{j=1}^m \Delta_{\mathbf{x}_j}\mathbf{e}_{\mathbf{x}_j}\mathbf{e}^\top_{\mathbf{x}_j}\right)\left(\bar{\mathscr{Y}}^t - \mu + \sum_{j=1}^m \delta_{\mathbf{x}_j}\mathbf{e}_{\mathbf{x}_j}\right),$$

Table 2: The flop analysis for the $i$th SMW iteration of sGMIA with PARDISO

| | |
|---|---|
| **SMW formula:** $(\bar{\mathbf{Q}}^{t+i})^{-1} = (\bar{\mathbf{Q}}^t)^{-1}(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\left(\mathbf{I}_{m\times m}+\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\right)^{-1}\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}$ | |
| I. **Preparation** $\mathbf{A}_i \triangleq \left(\mathbf{I}_{m\times m}+\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\right)^{-1}\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}$ | **Total:** $\leq 8i^2n+2in+2C_b+\frac{20}{3}i^2+2i$ |
| **1.** $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i = \bar{\mathbf{Q}}^t\backslash\mathbf{U}_i$ | $\leq 2C_b$ |
| **2.** $\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1} = \begin{bmatrix}\Delta_1 & & \\ & \ddots & \\ & & \Delta_m\end{bmatrix}\left((\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\right)^\top$ | $\leq 2in$ |
| **3.** $\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i$ | $\leq 4i^2$ |
| **4.** $\mathbf{I}_{m\times m}+\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i$ | $\leq 2i$ |
| **5.** LU decomposition: $\mathbf{I}_{m\times m}+\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i = \mathbf{LU}$ | $\leq \frac{8}{3}i^2$ |
| **6.** $\mathbf{L}^\top\backslash\left(\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\right)$ | $\leq (2i)^2n$ |
| **7.** $\mathbf{U}^\top\backslash\left(\mathbf{L}^\top\backslash\left(\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\right)\right)$ | $\leq (2i)^2n$ |
| II. **Conditional variance** $\mathrm{diag}(\bar{\mathbf{Q}}^{t+i})^{-1}$ | **Total:** $\leq 4in$ |
| **1.** Compute the diagonal of $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\mathbf{A}_i$ | $\leq 4in-n$ |
| **2.** $\mathrm{diag}(\bar{\mathbf{Q}}^{t+i})^{-1} = \mathrm{diag}(\bar{\mathbf{Q}}^t)^{-1} - \mathrm{diag}\left((\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\mathbf{A}_i\right)$ | n |
| III. **Conditional covariance** $(\bar{\mathbf{Q}}^{t+i})^{-1}\mathbf{e}_{\bar{\mathbf{x}}}$ | **Total:** $\leq 4in$ |
| **1.** $\mathbf{A}_i\mathbf{e}_{\bar{\mathbf{x}}}$ | 0 |
| **2.** $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i(\mathbf{A}_i\mathbf{e}_{\bar{\mathbf{x}}})$ | $\leq 4in-n$ |
| **3.** $(\bar{\mathbf{Q}}^{t+i})^{-1}\mathbf{e}_{\bar{\mathbf{x}}} = (\bar{\mathbf{Q}}^t)^{-1}\mathbf{e}_{\bar{\mathbf{x}}} - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i(\mathbf{A}_i\mathbf{e}_{\bar{\mathbf{x}}})$ | $n$ |
| IV. **Conditional mean** | **Total:** $\leq 4in+n+4in_2+C_b+10i$ |
| **1.** $\mathbf{b}_i \triangleq \begin{bmatrix}\Delta_1 & & \\ & \ddots & \\ & & \Delta_m\end{bmatrix}(\bar{y}-\mu)+\mathbf{Q}_\varepsilon^t\begin{bmatrix}\delta_1\\ \vdots \\ \delta_m\end{bmatrix}+\begin{bmatrix}\Delta_1 & & \\ & \ddots & \\ & & \Delta_m\end{bmatrix}\begin{bmatrix}\delta_1\\ \vdots \\ \delta_m\end{bmatrix}$ | $\leq 10i$ |
| **2.** $\mathbf{Q}_\varepsilon^t(\bar{\mathscr{Y}}-\mu)+\mathbf{b}_i$ | $\leq 2i$ |
| **3.** $\mathbf{A}_i\left(\mathbf{Q}_\varepsilon^t(\bar{\mathscr{Y}}-\mu)+\mathbf{b}_i\right)$ | $\leq 4in_2-2i$ |
| **4.** $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\left(\mathbf{A}_i\left(\mathbf{Q}_\varepsilon^t(\bar{\mathscr{Y}}-\mu)+\mathbf{b}_i\right)\right)$ | $\leq 4in-n$ |
| **5.** $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{b}_i = \bar{\mathbf{Q}}^t\backslash\mathbf{b}_i$ | $C_b$ |
| **6.** $\bar{\mu}^{t+i} = \bar{\mu}^t + (\bar{\mathbf{Q}}^t)^{-1}\mathbf{b}_i - (\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\left(\mathbf{A}_i\left(\mathbf{Q}_\varepsilon^t(\bar{\mathscr{Y}}-\mu)+\mathbf{b}_i\right)\right)$ | $2n$ |
| **Total computation cost:** $\leq 8i^2n+14in+n+4in_2+3C_b+\frac{20}{3}i^2+12i$ flops | |

where $\delta$s track the changes in $\bar{\mathscr{Y}}$ from $t$th iteration. Similar to (10), we define $\mathbf{b}_i$ to satisfy $\left(\mathbf{Q}_\varepsilon^t+\sum_{j=1}^m\Delta_{\mathbf{x}_j}\mathbf{e}_{\mathbf{x}_j}\mathbf{e}_{\mathbf{x}_j}^\top\right)\left(\bar{\mathscr{Y}}^t-\mu+\sum_{j=1}^m\delta_{\mathbf{x}_j}\mathbf{e}_{\mathbf{x}_j}\right)=\mathbf{Q}_\varepsilon^t(\bar{\mathscr{Y}}^t-\mu)+\mathbf{b}_i$. Note that $\mathbf{b}_i$ is a vector of $m$ nonzero entries corresponding to $\mathbf{x}_1,\ldots,\mathbf{x}_m$ and all other entries are 0, which again simplifies the conditional mean update as in the $(t+1)$th iteration.

The computation details and the corresponding costs at the $i$th SMW iteration after the last full iteration are summarized in Table 2. Note that the total computation cost is increasing in $i$, which implies the benefit of the SMW formula diminishes as SMW iterations continue after the last full iteration. The computation order in Table 2 is a result of careful flop analyses. For instance, there are several ways to compute $\mathbf{A}_i$ in Step I and we present the most efficient order of linear algebraic operations to compute $\mathbf{A}_i$ in I.1–I.7. Therefore, it is important to follow the order of computation outlined in Table 2 to maximize efficiency.

To summarize, sGMIA repeats one expensive full iteration and several cheaper SMW iterations. Suppose we perform the full iteration every $p$ iterations, where $p$ is a constant much smaller than $n$. Then, the

per-iteration computational cost of sGMIA is

$$\begin{aligned}&\frac{1}{p}\left(C_f+C_d+2C_b+n+\sum_{i=1}^{p-1}(8i^2n+14in+n+3C_b+4in_2+\frac{20}{3}i^2+12i)\right)\\=&\frac{(C_f+C_d-C_b)}{p}+\left(\frac{8}{3}n+\frac{20}{9}\right)p^2+\left(3n+2n_2+\frac{8}{3}\right)p-\frac{14}{3}n+3C_b-2n_2-\frac{44}{9}.\end{aligned}\tag{11}$$

Because $(C_f+C_d-C_b)/p\approx\mathscr{O}(n^{1.4})$, this is the leading order of (11). The rest of (11) is linear in $n$. Semelhago et al. (2020) show that the per-iteration computational cost of rGMIA is

$$\frac{(C_f+C_d+2C_b+6n)}{p}+\frac{7}{3}p^3+\frac{8}{3}p^2+(4n+8)p+7n+C_b+1,\tag{12}$$

if $|\mathscr{S}|=p$, which is the choice recommended by the authors. Notice that (12) has the same leading order as (11). However, sGMIA computes CEIs at all feasible solutions every iteration so that its search progress is not compromised from that of the original GMIA, whereas rGMIA only computes the CEI in $\mathscr{S}$ and restricts its search within $\mathscr{S}$ during rapid search.

The version of sGMIA without PARDISO is presented in Tables 3–4. The full iteration (Table 3) now computes the entire inverse of the sparse matrix $\bar{\mathbf{Q}}^t$ whose cost is denoted by $C_i$. From a Matlab time study, $C_i$ is estimated to be $\mathscr{O}(n^{2.2})$ for a two-dimensional problem. Table 4 only contains the steps of a SMW iteration different from those in Table 2. The per-iteration computational cost of sGMIA without PARDISO is

$$\begin{aligned}&\frac{1}{p}\left(C_i+2n_2n+n_2+\sum_{i=1}^{p-1}(8i^2n+18in+4in_2+\frac{20}{3}i^2+12i)\right)\\=&\frac{C_i+2n_2n+n_2}{p}+\left(\frac{8}{3}n+\frac{20}{9}\right)p^2+\left(n+2n_2+\frac{8}{3}\right)p-\frac{23}{3}n-2n_2-\frac{44}{9}.\end{aligned}$$

Again, the leading order, $\mathscr{O}(n^{2.2})$, is from the full iteration.

Note that with PARDISO, the per-iteration computational costs of both sGMIA and rGMIA are cheaper than that of a continuous GP update in (4).

Table 3: The flop analysis for a full iteration of sGMIA without PARDISO

| | |
|---|---|
| I. **Computing inverse** $(\bar{\mathbf{Q}}^t)^{-1}=\text{inv}(\bar{\mathbf{Q}}^t)$ | $C_i$ |
| II. **Conditional variance and covariance** $\text{diag}(\bar{\mathbf{Q}}^t)^{-1}$ and $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{e}_{\bar{\mathbf{x}}}$ | 0 |
| III. **Conditional mean** $\bar{\mu}^t=\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}+(\bar{\mathbf{Q}}^t)^{-1}\begin{bmatrix}\mathbf{0}\\ \mathbf{Q}^t_{\mathcal{E}}(\bar{\mathscr{Y}}_2-\mu_2)\end{bmatrix}$ | $2n_2n+n_2$ |
| **Total computation cost:** $C_i+2n_2n+n_2$ flops | |

Table 4: Changes in the $i$th SMW iteration of sGMIA without PARDISO

| | |
|---|---|
| I. **Preparation** $\mathbf{A}_i\triangleq\left(\mathbf{I}_{m\times m}+\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i\right)^{-1}\mathbf{V}_i^\top(\bar{\mathbf{Q}}^t)^{-1}$ | **Total:** $\le 8i^2n+2in+\frac{20}{3}i^2+2i$ |
| **1.** $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{U}_i$ | 0 |
| IV. **Conditional mean** | **Total:** $\le 8in+4in_2+10i$ |
| **5.** $(\bar{\mathbf{Q}}^t)^{-1}\mathbf{b}_i$ | $\le 4in-n$ |
| **Total computation cost:** $\le 8i^2n+18in+4in_2+\frac{20}{3}i^2+12i$ flops | |

## 5 ADAPTIVE PARAMETER SELECTION

A question remains to determine the best value of $p$ to minimize the per-iteration computational cost of sGMIA. For sGMIA with PARDISO, $p$ minimizing the overall order of (11) is $p \approx \mathcal{O}(n^{0.4/3})$. However, this does not provide a practical guidance on choosing $p$. Instead, we implement the following empirical method to adaptively adjust the length of SMW iterations.

Let $t_0$ be the computation time of a full iteration, and $t_1, t_2, \ldots, t_{p-1}$ be those of the next $p-1$ SMW iterations. Then, the per-iteration computational cost is minimized, if $p$ satisfies

$$\frac{t_0+t_1+\cdots+t_{p-1}}{p} < \frac{t_0+t_1+\cdots+t_{p-1}+t_p}{p+1} \iff t_p > \frac{1}{p}(t_0+t_1+\cdots+t_{p-1}). \tag{13}$$

That is, the cost of the next SMW iteration is larger than the current per-iteration average.

Although $t_0, t_1 \ldots, t_{p-1}$ can be measured on the fly, the next-iteration time, $t_p$, is unknown a priori. Exploiting that the overall cost of the $i$th SMW iteration is quadratic in $i$ from Tables 2 and 4, we fit a quadratic regression model in $i$ for $t_i$ after running a reasonably large number of SMW iterations, and use the predicted $t_p$ in (13). We quit the SMW iterations when the inequality (13) is satisfied and perform a full iteration.

## 6 EXPERIMENT RESULTS

In this section, we use the ($s$, $S$) inventory problem from Koenig and Law (1985) to contrast the performances of sGMIA and rGMIA. We choose the decision variables to be ($x_1$, $x_2$) = ($s$, $S-s$), and the objective function is the expected value of the cost over a 30-day period. The feasible region for $(x_1, x_2)$ is set to be $\mathscr{X} = [1, \ldots, 100] \times [1, \ldots, 100]$. All other parameters are identical to those used in Salemi et al. (2019). Based on 500,000 replications of simulation at each feasible solution, the optimal solutions is estimated to be $(x_1, x_2) = (17, 36)$ with expected cost \$106.14.

To estimate the initial parameteres of the GMRF, we apply the generalized method of moments (GMM) approach by Song and Dong (2018). For all experiments below, we choose 10 initial central design points by Latin hypercube sampling and 80 additional design points are selected by the slim-GMM algorithm in Song and Dong (2018). At all design points, 10 replications were run to estimate the parameters. Song and Dong (2018) show that the computation time for GMM does not depend on $n$, which makes it much faster than maximum likelihood estimation when $n$ is large. Moreover, the GMM estimators tend to be more stable than maximum likelihood estimators when the initial sample size is small to moderate.

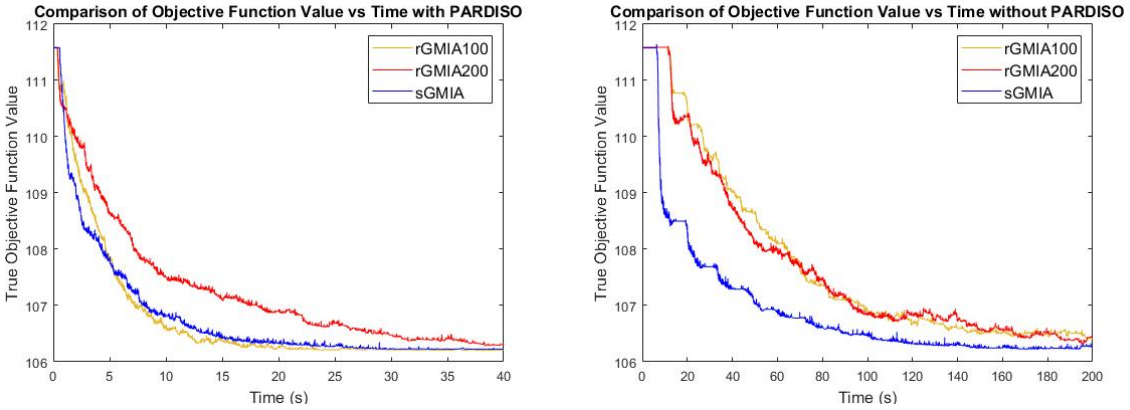
In the following sections, we compare sGMIA and rGMIA in terms of their computational time as well as search progress. For rGMIA, we test two choices of $p$, $p = 100$ and 200, and set $|\mathscr{S}| = p$. For sGMIA, to determine the length of SMW iterations in each period, we fit a quadratic regression model in $i$ for $t_i$. For sGMIA with PARDISO, we first fit a model after 20 SMW iterations. For the version without PARDISO, $p$ tends to be larger as a full iteration is more expensive. Therefore, the regression model is fitted after 100 SMW iterations. We do not refit the regression within the period unless the relative difference between predicted time and the actual time exceeds 0.2. Both sGMIA and rGMIA are run with and without PARDISO. All experiments are run on a single thread of Intel® Core™i5-9600K CPU (3.70GHz) with 16.0 GB RAM.

### 6.1 COMPUTATIONAL TIME COMPARISON OF sGMIA AND rGMIA

Table 5 shows the average time of full/global-search and SMW/rapid-search iterations, and the per-iteration computational cost for both sGMIA and rGMIA with or without PARDISO. Notice that per-iteration time of sGMIA is longer than that of rGMIA across all settings. However, considering that sGMIA computes CEIs at all solutions every iteration, we argue that the computation time of sGMIA is quite comparable to that of rGMIA, especially without PARDISO. Also, notice that efficiency of rGMIA depends heavily on the choice of $p$, however, it is difficult to choose the optimal $p$ prior to running simulations.

Table 5: Average computation time obtained from 50 macro runs of 2,000 iterations of sGMIA and rGMIA applied to the $(s,S)$ inventory problem. Standard errors are provided in parentheses.

| | With PARDISO | | | Without PARDISO | | |
|---|---|---|---|---|---|---|
| Algorithm | sGMIA | rGMIA | | sGMIA | rGMIA | |
| $p$ value | Adaptive | $p=100$ | $p=200$ | Adaptive | $p=100$ | $p=200$ |
| Full (Global-search) Iteration Time (s) | 0.431 (0.005) | 0.186 (0.004) | 0.327 (0.009) | 6.754 (0.047) | 6.889 (0.063) | 7.037 (0.131) |
| SMW (Rapid-search) Iteration Time (s) | 0.010 (0.000) | 0.004 (0.000) | 0.008 (0.000) | 0.037 (0.000) | 0.004 (0.000) | 0.011 (0.001) |
| Per-iteration Time (s) | 0.012 (0.000) | 0.006 (0.000) | 0.009 (0.000) | 0.079 (0.001) | 0.076 (0.001) | 0.050 (0.001) |




(a) With PARDISO (b) Without PARDISO

Figure 1: The trajectory of the objective function value at the current best plotted against run time averaged across 50 macro runs.

For sGMIA with PARDISO, the mode of $p$ values chosen adaptively throughout 50 macro runs is 43. We noticed $p$ is more variable with PARDISO than without PARDISO. In the latter case, the average $p$ is 170.5 with standard deviation 2.6.

### 6.2 SEARCH PROGRESS COMPARISON OF sGMIA AND rGMIA

Figure 1 presents the trajectory of the objective function at the current best solution found by each algorithm against its run time. Each curve is a time average of 50 macro runs of each algorithm. Figure 1(a) displays the case with PARDISO. Both sGMIA and rGMIA with $p=100$ exhibit similar search progresses, whereas rGMIA with $p=200$ is the slowest among three algorithms, which again shows that the performance of rGMIA is quite sensitive to the choice of $p$. Figure 1(b) shows performances of the algorithms without PARDISO. Notice that the search progress of sGMIA is much faster than both settings of rGMIA.

## 7 CONCLUSIONS

In this paper, we propose sGMIA that achieves the same search progress as GMIA with significantly reduced computational cost. Rigorous flop analyses show that the per-iteration computational cost of sGMIA with PARDISO is smaller than that of a continuous GP-based algorithm and of the same order as rGMIA. We also provide a version of sGMIA without PARDISO. Another salient feature of sGMIA is that it monitors computational cost of each iteration and adaptively chooses its parameter to achieve the

smallest per-iteration computational cost. The empirical results demonstrate competitiveness of sGMIA in both computation time and search progress.

## ACKNOWLEDGMENTS

This research is supported by NSF DMS-1854659 and the Institute of Cyberscience seed grant program at the Pennsylvania State University.

## REFERENCES

Frazier, P. I. 2018. "A Tutorial on Bayesian Optimization". https://arXiv:1807.02811, accessed 10th February 2020.

Frazier, P. I., J. Xie, and S. E. Chick. 2011. "Value of Information Methods for Pairwise Sampling with Correlations". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 3974–3986. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Jones, D. R., M. Schonlau, and W. J. Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions". *Journal of Global Optimization* 13(4):455–492.

Koenig, L. W., and A. M. Law. 1985. "A Procedure for Selecting a Subset of Size $m$ Containing the $l$ Best of $k$ Independent Normal Populations. with Applications to Simulation". *Communications in Statistics* B14(3):719–734.

Petra, C. G., O. Schenk, and M. Anitescu. 2014. "Real-Time Stochastic Optimization of Complex Energy Systems on High-Performance Computers". *Computing in Science Engineering* 16(5):32–42.

Petra, C. G., O. Schenk, M. Lubin, and K. Gäertner. 2014. "An Augmented Incomplete Factorization Approach for Computing the Schur Complement in Stochastic Optimization". *SIAM Journal on Scientific Computing* 36(2):C139–C162.

Rasmussen, C., and C. Williams. 2006. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. MA: The MIT Press.

Rue, H., and L. Held. 2005. *Gaussian Markov Random Fields: Theory and Applications*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. New York: Chapman and Hall/CRC.

Salemi, P., E. Song, B. L. Nelson, and J. Staum. 2019. "Gaussian Markov Random Fields for Discrete Optimization via Simulation: Framework and Algorithms". *Operations Research* 67(1):250–266.

Semelhago, M., B. L. Nelson, E. Song, and A. Wächter. 2020. "Rapid Discrete Optimization via Simulation with Gaussian Markov Random Fields". To appear in INFORMS Journal on Computing.

Semelhago, M., B. L. Nelson, A. Wächter, and E. Song. 2017. "Computational Methods for Optimization via Simulation Using Gaussian Markov Random Fields". In *Proceedings of 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 2080–2091. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Song, E., and Y. Dong. 2018. "Generalized Method of Moments Approach to Hyperparameter Estimation for Gaussian Markov Random Fields". In *Proceedings of 2018 Winter Simulation Conference*, edited by M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 1790–1801. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sun, L., L. J. Hong, and Z. Hu. 2014. "Balancing Exploitation and Exploration in Discrete Optimization via Simulation Through a Gaussian Process-Based Search". *Operations Research* 62(6):1416–1438.

Takahashi, K., J. Fagan, and C. M. S.. 1973. "Formation of a Sparse Bus Impedance Matrix and its Application to Short Circuit Study". In *PICA Conference Proceedings*, 63–69: New York : Institute of Electrical and Electronics Engineers.

Xie, J., P. I. Frazier, and S. E. Chick. 2016. "Bayesian Optimization via Simulation with Pairwise Sampling and Correlated Prior Beliefs". *Operations Research* 64(2):542–559.

## AUTHOR BIOGRAPHIES

**XINRU LI** is a PhD student in the Department of Industrial and Manufacturing Engineering at the Penn State University. Her research interests include simulation optimization, and computer experiments. Her email address is xul277@psu.edu and her website can be found at https://sites.google.com/view/xinruli.

**EUNHYE SONG** is the Harold and Inge Marcus Early Career Assistant Professor in the Department of Industrial and Manufacturing Engineering at the Penn State University. Her research interests include simulation design of experiments, simulation uncertainty and risk quantification, optimization via simulation under input model risk and large-scale discrete optimization via simulation. She has been serving on the INFORMS-Simulation Diversity committee since 2018. Her email address is eus358@psu.edu and her website can be found at http://eunhyesong.info.