Surrogate-Based Simulation Optimization

L. Jeff Hong
School of Management and School of Data Science, Fudan University, Shanghai 200433, China, hong_liu@fudan.edu.cn

Xiaowei Zhang
Faculty of Business and Economics, The University of Hong Kong, Pokfulam Road, Hong Kong SAR, xiaoweiz@hku.hk

Simulation models are widely used in practice to facilitate decision-making in a complex, dynamic and stochastic environment. But they are computationally expensive to execute and optimize, due to lack of analytical tractability. Simulation optimization is concerned with developing efficient sampling schemes—subject to a computational budget—to solve such optimization problems. To mitigate the computational burden, surrogates are often constructed using simulation outputs to approximate the response surface of the simulation model. In this tutorial, we provide an up-to-date overview of surrogate-based methods for simulation optimization with continuous decision variables. Typical surrogates, including linear basis function models and Gaussian processes, are introduced. Surrogates can be used either as a local approximation or a global approximation. Depending on the choice, one may develop algorithms that converge to either a local optimum or a global optimum. Representative examples are presented for each category. Recent advances in large-scale computation for Gaussian processes are also discussed.

Key words: Surrogate, simulation optimization, Gaussian process, matrix inversion

1. Introduction

Simulation optimization (SO) concerns a class of optimization problems whose objective functions and/or constraints do not possess an analytical form and can only be evaluated based on noisy simulation samples. The simulation model is usually expensive to execute; thus, the number of evaluations that one is allowed to perform is limited, subject to one's computational budget. Specifically, we consider in this tutorial problems of the form

$$\max_{\mathbf{x}\in\mathcal{X}}\{f(\mathbf{x}) \coloneqq \mathbb{E}[F(\mathbf{x})]\}, \tag{1}$$

where $\mathbf{x}$ is the decision variable, $\mathcal{X}$ is the feasible set, and $F(\mathbf{x})$ is a real-valued random variable, representing the stochastic response of a simulation model evaluated at $\mathbf{x}$. The distribution of $F(\mathbf{x})$ is an unknown function of $\mathbf{x}$, but its samples can be generated from running simulation experiments.

Depending on the nature of the feasible set, problem (1) demands a distinct treatment and principle for designing algorithms. When $\mathcal{X}$ is a set of a relatively small number of feasible solutions

with no inherent ordering defined, the problem is known as *ranking and selection* (R&S). For example, $\mathcal{X}$ may represent feasible system configurations with regard to how many and what redundant components to use to design a reliable system.

Another common setting is that $\mathcal{X}$ is integer-ordered, that is, $\mathcal{X} = \Omega \cap \mathbb{Z}^d$, where $\Omega \subset \mathbb{R}^d$ is a convex set and $\mathbb{Z}^d$ is the set of d-dimensional integer vectors. In this setting, $\mathcal{X}$ usually has a very large or even infinite number of elements, and problem (1) is called *discrete optimization via simulation* (DOvS). For example, a retailer may need to make stocking decisions for d products to minimize operational costs, $\mathbf{x}$ may represent the numbers of units to order for these products, and $\mathcal{X}$ may be formed by capacity constraints.

In this tutorial, we assume that $\mathbf{x}$ is a d-dimensional real vector and $\mathcal{X} \subseteq \mathbb{R}^d$ is a continuous set, and refer to Hong et al. (2015) for a recent tutorial on R&S and DOvS problems. However, our concentration on continuous decision variables does not limit the scope of this tutorial. In general, algorithms designed for the continuous setting can be applied to DOvS, although theoretical analyses may need to be adjusted.

A variety of strategies have been proposed to solve continuous SO problems, including sample average approximation (Kim et al. 2015), stochastic approximation (Chau and Fu 2015), and random search (Andradóttir 2015). This tutorial focuses on surrogate-based methods—another class of strategies—which have gained increasing interest in recent years, despite their long history. The popular use of surrogates in SO may be attributed to (i) the flexibility to capture complex surface shapes and (ii) the capability to predict surface values where no simulation samples are observed. The latter reason is of particular importance in light of the fact that computational budget is generally regarded as a scarce resource relative to the cost of simulation experiments.

Finally, we note that in contrast to earlier articles on surrogate-based methods (Barton and Meckesheimer 2006, Barton 2009), the present tutorial aims to bring the readers up to date on the fast developments in the area. One featured discussion is on recent advances in coping with the computational challenges associated with the use of surrogates for large-scale problems.

The rest of this tutorial is organized as follows. Section 2 introduces surrogates that are widely used in practice. Section 3 and Section 4 present SO methods that use surrogates as local approximations and global approximations, respectively. Section 5 discusses recent advances in handling computational issues that arise when using surrogates for large datasets. Section 6 highlights current research challenges and potential opportunities.

2. Surrogates

A surrogate—also known as (a.k.a.) metamodel—is an approximation to the response surface, that is, the simulation input-output relationship. The main purpose of using a surrogate is to

mitigate the computational burden of running expensive simulation experiments. Although any supervised learning (Hastie et al. 2009) model may, in principle, be used, one typically has several considerations to keep in mind when choosing a surrogate to cope with computational budget constraints.

(i) It should possess a simple structure and does not require "big data" to fit, because simulation samples are expensive to acquire.

(ii) It should be computationally easy to fit, because it often needs to be updated in a sequential fashion as more simulation samples become available.

(iii) It should give rise to a predictor in explicit form, so that predictions can be computed efficiently, theoretical analysis can be facilitated, and the surrogate can be optimized easily.

Three classes of surrogates that satisfy the above criteria have been widely adopted in practice: low-order polynomials, linear basis function models, and Gaussian processes; we introduce them in Sections 2.1, 2.2, and 2.3, respectively. The first—including linear and quadratic functions—are normally viewed as local approximations from the perspective of Taylor expansion, whereas the last two are global approximations thanks to their nonparametric nature. Nevertheless, as we demonstrate in Section 2.4, they can be unified through the lens of ridge regularization. In Section 2.5, we present recent approaches to enhancing the prediction capability of a surrogate if additional information is available.

Before formally presenting the surrogates, we state the common set-up. Suppose that the simulation model is executed at $\{\mathbf{x}_i : i = 1, \ldots, n\}$, where $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$ for each i. For each $\mathbf{x}_i$, the number of replications is $r_i \geq 1$; each replication generates a realization of the random variable $F(\mathbf{x}_i)$ in Equation (1), denoted by $y_{i,\ell}$ for $\ell = 1, \ldots, r_i$. Let $y_{i,\ell} = f(\mathbf{x}_i) + \epsilon_{i,\ell}$, where $\epsilon_{i,\ell}$ is independent Gaussian noise with mean zero and variance $\sigma^2(\mathbf{x}_i)$. We are interested in approximating f via the regression equation

$$\bar{y}_i = f(\mathbf{x}_i) + \bar{\epsilon}_i, \quad i = 1, \ldots, n, \tag{2}$$

where $\bar{y}_i \coloneqq {r_i}^{-1} \sum_{\ell=1}^{r_i} y_{i,\ell}$ and $\bar{\epsilon}_i \coloneqq {r_i}^{-1} \sum_{\ell=1}^{r_i} \epsilon_{i,\ell}$. Clearly, $\mathrm{Var}[\bar{\epsilon}_i] = \sigma^2(\mathbf{x}_i)/r_i$.

2.1. Polynomials

Due to the explosion in the number of terms in the representation of a polynomial in multiple dimensions, polynomials with orders higher than two are seldom used to approximate a response surface. Low-order polynomials are suitable for situations where we are interested in a localized region of the feasible set (or design space) $\mathcal{X}$. A second-order polynomial (i.e., quadratic function) is

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j + \sum_{j=1}^{d} \sum_{k=1}^{d} \beta_{jk} x_j x_k, \tag{3}$$

where $\mathbf{x} = (x_1, \ldots, x_d)$. This surrogate may be appropriate if we expect the response surface f has substantial curvature. In contrast, a first-order polynomial (i.e., linear function), to which Equation (3) is reduced by setting $\beta_{jk} = 0$ for all j and k, may be a better fit in the presence of little curvature.

The parameters $\beta_0, \beta_j, \beta_{jk}$ can be estimated via ordinary least squares (OLS)

$$\min_{\beta_0,\beta_j,\beta_{jk}} \frac{1}{n}\sum_{i=1}^{n}\left(\bar{y}_i - \beta_0 - \sum_{j=1}^{d}\beta_j x_{i,j} - \sum_{j=1}^{d}\sum_{k=1}^{d}\beta_{jk}x_{i,j}x_{i,k}\right)^2. \tag{4}$$

The solution is given in Section 2.2 in a more general setting. Then, one can predict the response at any arbitrary location $\mathbf{x}$ by plugging the OLS estimates $\hat{\beta}_0, \hat{\beta}_j, \hat{\beta}_{jk}$ in Equation (3), that is,

$$\hat{f}(\mathbf{x}) = \hat{\beta}_0 + \sum_{j=1}^{d}\hat{\beta}_j x_j + \sum_{j=1}^{d}\sum_{k=1}^{d}\hat{\beta}_{jk}x_j x_k.$$

2.2. Linear Basis Function Models

If low-order polynomials do not provide a good fit, a natural extension uses linear basis function models that express f as a linear combination of basis functions,

$$f(\mathbf{x}) = \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x}) = \sum_{k=1}^{p}\beta_k\phi_k(\mathbf{x}), \tag{5}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\intercal$ is a vector of unknown parameters and $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_p(\mathbf{x}))^\intercal$ is a vector of chosen basis functions, such as the truncated power basis and the Fourier basis, etc.; see Bishop (2006, Chapter 3) and Hastie et al. (2009, Chapter 5).

A particular popular class, among others, is radial basis functions (RBFs), meaning that the value of ϕ depends on $\mathbf{x}$ only through the distance between $\mathbf{x}$ and some fixed point, say $\mathbf{c} \in \mathbb{R}^d$. That is, an RBF has the form $\varphi(\|\mathbf{x} - \mathbf{c}\|)$ for some function $\varphi : \mathbb{R} \mapsto \mathbb{R}$. Typical examples include the Gaussian RBF $\varphi(x) = \exp(-x^2/2\eta^2)$ and the thin plate spline $\varphi(x) = x^2\ln(x)$.

The parameters $\boldsymbol{\beta}$ can be also estimated via OLS:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}(\bar{y}_i - \boldsymbol{\beta}^\intercal\boldsymbol{\phi}(\mathbf{x}_i))^2 \\ &= \boldsymbol{\Phi}^\intercal(\boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal)^{-1}\bar{\mathbf{y}},\end{aligned} \tag{6}$$

where $\boldsymbol{\Phi}$ is the n-by-p matrix with the i-th row being $\boldsymbol{\phi}(\mathbf{x}_i)^\intercal$ for all $i = 1, \ldots, n$. The prediction is then given by

$$\hat{f}(\mathbf{x}) = \hat{\boldsymbol{\beta}}^\intercal\boldsymbol{\phi}(\mathbf{x}) = (\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}))^\intercal(\boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal)^{-1}\bar{\mathbf{y}}. \tag{7}$$

2.3. Gaussian Processes

Gaussian processes (GPs, a.k.a. Gaussian random fields) have a remarkable success in numerous research areas. Using them as surrogates originated in geostatistics, where the method was named *kriging* (Krige 1951, Matheron 1963). Later, they were applied to the design and analysis of (deterministic) computer experiments (Sacks et al. 1989). The adoption of GPs to approximate response surfaces in stochastic simulation literature, where the method is often called *stochastic kriging*, was popularized by Ankenman et al. (2010). The same method was referred to as *GP regression* in the machine learning literature (Rasmussen and Williams 2006).

Whereas linear regression with basis functions is a frequentist approach, GPs represent a Bayesian viewpoint. A GP with domain $\mathcal{X}$ is fully characterized by its mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and covariance function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. To use a GP as a surrogate, we start by imposing a GP prior on the unknown function f, denoted by $f \sim \mathsf{GP}(\mu, K)$. Some may prefer to state the assumption in a different way—with a somewhat less Bayesian flavor—namely, f is a sample path (i.e., realization) of the GP. Both mean the following: for any finite set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathcal{X}$ of any size $n \geq 1$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$ has a multivariate normal distribution with mean vector $\boldsymbol{\mu} = (\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_n))^\intercal$ and n-by-n covariance matrix $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_{i'}))_{i,i'=1}^n$.

2.3.1. Mean Functions From a modeling point of view, the mean function μ is used to encode one's prior knowledge about the overall shape of the response surface f. It is usually chosen in one of the following ways.

(i) Set $\mu(\mathbf{x}) \equiv c$ for some constant c representing the overall surface mean. This is possibly the most common treatment in practice. One may even set $c = 0$ if little prior knowledge is available.

(ii) Set $\mu(\mathbf{x}) = \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ is a vector of known basis functions and $\boldsymbol{\beta}$ is a vector of hyperparameters of compatible dimension.

(iii) Set $\mu(\mathbf{x})$ to be a function derived from a simplified, analytical model of the same stochastic system that the original simulation model aims to describe. For example, if the original simulation model is a complex queueing model, μ may be derived in closed form based on a highly stylized queueing model; see Section 2.5 for details.

2.3.2. Covariance Functions We first introduce two popular classes of covariance functions: the Gaussian class and the Matérn class. They are both *stationary*, meaning that $\mathrm{Cov}[f(\mathbf{x}), f(\mathbf{x}')]$ depends on $\mathbf{x}$ and $\mathbf{x}'$ only through the difference $(\mathbf{x} - \mathbf{x}')$. We then present a class of covariance functions that permits a different level of differentiability in each dimension. More examples of covariance functions can be found in Rasmussen and Williams (2006, Chapter 4).

Example 1 (Gaussian Covariance Functions). For positive constants τ and η, a Gaussian covariance function is defined by

$$K_{\mathsf{Gaussian}}(\mathbf{x},\mathbf{x}') = \tau^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\eta^2}\right), \quad \mathbf{x},\mathbf{x}' \in \mathbb{R}^d. \tag{8}$$

It is also called a squared exponential covariance function.

Example 2 (Matérn Covariance Functions). For positive constants τ, η, and ν, a Matérn covariance function is defined by

$$K_{\mathsf{Matern}}(\mathbf{x},\mathbf{x}';\nu) = \frac{\tau^2}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}\|\mathbf{x}-\mathbf{x}'\|}{\eta}\right)^{\nu} \mathsf{K}_{\nu}\left(\frac{\sqrt{2\nu}\|\mathbf{x}-\mathbf{x}'\|}{\eta}\right), \quad \mathbf{x},\mathbf{x}' \in \mathbb{R}^d, \tag{9}$$

where $\Gamma(\cdot)$ is the gamma function, and $\mathsf{K}_{\nu}(\cdot)$ is the modified Bessel function of the second kind of order ν. The parameter ν is usually set to be half-integer, i.e., $1/2, 3/2, 5/2, \ldots$, in which case the expression of $K_{\mathsf{Matern}}(\mathbf{x},\mathbf{x}';\nu)$ can be simplified substantially. For instance,

$$K_{\mathsf{Matern}}(\mathbf{x},\mathbf{x}';\nu) = \begin{cases} \tau^2 \exp\left(\dfrac{-\|\mathbf{x}-\mathbf{x}'\|}{\eta}\right), & \text{if } \nu = 1/2, \\ \tau^2 \left(1+\dfrac{\sqrt{3}\|\mathbf{x}-\mathbf{x}'\|}{\eta}\right) \exp\left(-\dfrac{\sqrt{3}\|\mathbf{x}-\mathbf{x}'\|}{\eta}\right), & \text{if } \nu = 3/2, \\ \tau^2 \left(1+\dfrac{\sqrt{5}\|\mathbf{x}-\mathbf{x}'\|}{\eta}+\dfrac{5\|\mathbf{x}-\mathbf{x}'\|^2}{3\eta^2}\right) \exp\left(-\dfrac{\sqrt{5}\|\mathbf{x}-\mathbf{x}'\|}{\eta}\right), & \text{if } \nu = 5/2. \end{cases}$$

A general formula can be found on page 85 of Rasmussen and Williams (2006).

Note that both the expressions in (8) and (9) are in the form of $\tau^2\rho(\|\mathbf{x}-\mathbf{x}'\|)$ for some function $\rho : \mathbb{R}_{\geq 0} \mapsto (0,1]$. Thus, the parameter τ^2 represents the marginal variance of the associated GP and $\rho(\|\mathbf{x}-\mathbf{x}'\|)$ represents the correlation.

The parameter ν of the Matérn covariance functions is called the *smoothness* parameter, for it controls the order of differentiability of the sample paths of the induced GP; see Stein (1999, Section 6.5). It can be shown (Stein 1999, page 50) that

$$\lim_{\nu\to\infty} K_{\mathsf{Matern}}(\mathbf{x},\mathbf{x}';\nu) = K_{\mathsf{Gaussian}}(\mathbf{x},\mathbf{x}'), \quad \mathbf{x},\mathbf{x}' \in \mathbb{R}^d.$$

This suggests that the sample paths induced by the Gaussian covariance functions are infinitely differentiable—an overly strong property that may not be reasonable for some response surfaces.

Being controlled by a single parameter ν, the differentiability of the GPs associated with the Matérn covariance functions are homogeneous in each dimension. Motivated by the need for flexibility to control differentiability separately in different dimensions, Salemi et al. (2019a) propose a new class of covariance functions, and the corresponding GPs are called *generalized integrated Brownian fields* (GIBFs). Unlike the Gaussian and Matérn classes, the GIBF covariance functions are nonstationary and possess a tensor product form.

EXAMPLE 3 (GIBF COVARIANCE FUNCTIONS). For each $j = 1, \ldots, d$, let $m_j \geq 0$ be an integer and $\boldsymbol{\theta}_j = (\theta_{j,0}, \theta_{j,1}, \ldots, \theta_{j,m_j+1}) \in \mathbb{R}_{>0}^{m_j+2}$. A GIBF covariance function is defined by

$$\begin{aligned} K_{\mathsf{GIBF}}(\mathbf{x}, \mathbf{x}'; \mathbf{m}, \boldsymbol{\theta}) &= \prod_{j=1}^{d} K_j(x_j, x_j'; m_j, \boldsymbol{\theta}_j), \quad \mathbf{x}, \mathbf{x}' \in \mathbb{R}_{\geq 0}^d, \\ K_j(x_j, x_j'; m_j, \boldsymbol{\theta}_j) &= \sum_{\ell=0}^{m_j} \theta_{j,\ell} \frac{(x_j x_j')^\ell}{(\ell!)^2} + \theta_{j,m_j+1} \int_0^\infty \frac{(x_j - u)_+^{m_j} (x_j' - u)_+^{m_j}}{(m_j!)^2} \, du, \end{aligned} \tag{10}$$

where $\mathbf{m} = (m_1, \ldots, m_d)$ and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_d)$, and $(x)_+ = \max(x, 0)$ for $x \in \mathbb{R}$.

A notable property, among others, of an $\mathbf{m}$-GIBF is that its sample path is m_j times differentiable along the j-th coordinate, for each $j = 1, \ldots, d$. Moreover, if $m_j \geq 1$, then the derivative of an $\mathbf{m}$-GIBF with respect to the j-th coordinate is a $(m_1, \ldots, m_j - 1, \ldots, m_j + 1, \ldots, m_d)$-GIBF.

2.3.3. GP Regression If the simulation noise $\epsilon_{i,\ell}$ is independent of the GP prior and has a Gaussian distribution with a *known* variance, then the posterior distribution of f is also a GP. Let $\mathcal{D}_n \coloneqq \{(\mathbf{x}_i, \bar{y}_i) : i = 1, \ldots, n\}$ denote the simulation data. The posterior mean function and covariance functions are given by

$$\mu_n(\mathbf{x}) \coloneqq \mathbb{E}[f(\mathbf{x}) | \mathcal{D}_n] = \mu(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\intercal (\mathbf{K} + \boldsymbol{\Sigma})^{-1} (\bar{\mathbf{y}} - \boldsymbol{\mu}), \tag{11}$$

$$K_n(\mathbf{x}, \mathbf{x}') \coloneqq \mathrm{Cov}[f(\mathbf{x}), f(\mathbf{x}') | \mathcal{D}_n] = K(\mathbf{x}, \mathbf{x}') - \mathbf{k}(\mathbf{x})^\intercal (\mathbf{K} + \boldsymbol{\Sigma})^{-1} \mathbf{k}(\mathbf{x}'), \tag{12}$$

where $\mathbf{k}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n))^\intercal$, $\bar{\mathbf{y}} = (\bar{y}_1, \ldots, \bar{y}_n)^\intercal$, and $\boldsymbol{\Sigma}$ is the n-by-n diagonal matrix with the i-th diagonal element being $\sigma^2(\mathbf{x}_i)/r_i$. Then, one can simply use $\mu_n(\mathbf{x})$ to predict the response surface, that is,

$$\hat{f}(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\intercal (\mathbf{K} + \boldsymbol{\Sigma})^{-1} (\bar{\mathbf{y}} - \boldsymbol{\mu}). \tag{13}$$

Further, being the conditional expectation given the observations, $\hat{f}(\mathbf{x})$ is the best predictor that minimizes the mean squared prediction error; see Rice (2007, page 153).

2.3.4. Selection of Hyperparameters A *hyperparameter* is a parameter of a prior distribution in Bayesian statistics. In the context of GP regression, hyperparameters are those used to specify the mean function μ and the covariance function K. For example, if we choose $\mu(\mathbf{x}) = \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x})$ and $K(\mathbf{x}, \mathbf{x}') = \tau^2 \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\eta^2)$, then the hyperparameters are $(\boldsymbol{\beta}, \tau, \eta)$.

Let $\boldsymbol{\theta}$ denote the collection of hyperparameters. Let $\mathbf{X}$ denote the n-by-d matrix whose i-th row is $\mathbf{x}_i^\intercal$ for all $i = 1, \ldots, n$. A usual approach to selecting $\boldsymbol{\theta}$ is to maximize the following log "likelihood",

$$\ln \mathsf{p}(\bar{\mathbf{y}} | \mathbf{X}) = -\frac{1}{2} (\bar{\mathbf{y}} - \boldsymbol{\mu}(\boldsymbol{\theta}))^\intercal (\mathbf{K}(\boldsymbol{\theta}) + \boldsymbol{\Sigma})^{-1} (\bar{\mathbf{y}} - \boldsymbol{\mu}(\boldsymbol{\theta})) - \frac{1}{2} \ln |\mathbf{K}(\boldsymbol{\theta})| - \frac{n}{2} \ln(2\pi), \tag{14}$$

where $|\cdot|$ denotes the determinant of a square matrix, and we write $\boldsymbol{\mu}(\boldsymbol{\theta})$ and $\mathbf{K}(\boldsymbol{\theta})$ to stress the dependence of $\boldsymbol{\mu}$ and $\mathbf{K}$ on $\boldsymbol{\theta}$. This is a nonlinear optimization problem with possibly multiple local

optima. Gradients of $\ln \mathsf{p}(\bar{\mathbf{y}}|\mathbf{X})$ with respect to $\boldsymbol{\theta}$ can often be derived analytically and provided to numerical optimization algorithms such as L-BFGS; see Ankenman et al. (2010) for details.

A comment is warranted here regarding the notation of likelihood, however. In Bayesian statistics, the likelihood—also termed the sampling distribution—is referring to the distribution of the observed data conditional on the data-generating process. In the context of GP regression, the likelihood is $\mathsf{p}(\bar{\mathbf{y}}|\mathbf{X},\mathbf{f})$, where $\mathbf{f} = (f(\mathbf{x}_1),\dots,f(\mathbf{x}_n))^\intercal$. The conditioning on $\mathbf{f}$ is necessary because f is a *random* function or sample path of the GP prior. Note that $\ln \mathsf{p}(\bar{\mathbf{y}}|\mathbf{X},\mathbf{f})$ is *not* identical to Equation (14). Indeed,

$$\begin{aligned}\ln \mathsf{p}(\bar{\mathbf{y}}|\mathbf{X},\mathbf{f}) &= \ln \prod_{i=1}^{n} \frac{1}{\sigma(\mathbf{x}_i)\sqrt{2\pi}} \exp\left(-\frac{(\bar{y}_i - \mu(\mathbf{x}_i))^2}{2\sigma^2(\mathbf{x}_i)}\right) \\ &= -\frac{1}{2}\sum_{i=1}^{n}(\bar{\mathbf{y}} - \boldsymbol{\mu}(\boldsymbol{\theta}))^\intercal \boldsymbol{\Sigma}^{-1}(\bar{\mathbf{y}} - \boldsymbol{\mu}(\boldsymbol{\theta})) - \frac{n}{2}\ln(2\pi).\end{aligned}$$

Technically, $\mathsf{p}(\bar{\mathbf{y}}|\mathbf{X})$ is called the log *marginal likelihood* in Bayesian statistics, because it is the marginalization over $\mathbf{f}$:

$$\mathsf{p}(\bar{\mathbf{y}}|\mathbf{X}) = \int \mathsf{p}(\bar{\mathbf{y}}|\mathbf{X},\mathbf{f})\mathsf{p}(\mathbf{f}|\mathbf{X})\, d\mathbf{f},$$

where $\mathsf{p}(\mathbf{f}|\mathbf{X})$ is the prior of $\mathbf{f}$, which is a multivariate normal distribution; see Rasmussen and Williams (2006, Chapter 5) for more discussion.

2.4. A Connection via Ridge Regularization

We start with the linear basis function model (5) for the case that many basis functions are included, even to the point of overparameterization $p \gg n$. In this case, the OLS solution (6) is numerically unstable because the matrix $\boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal$ is nearly singular if p is smaller than but close to n and becomes singular if $p \geq n$. As a result, $\|\hat{\boldsymbol{\beta}}\|$—the Euclidean norm of $\hat{\boldsymbol{\beta}}$—would explode, and the prediction power of (7) would be poor.

We now add ridge regularization (Hastie 2020) to the least squares formulation to penalize the magnitude in norm of the solution, resulting in the method of regularized least squares (RLS):

$$\min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}(\bar{y}_i - \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x}_i))^2 + \lambda\|\boldsymbol{\beta}\|^2, \tag{15}$$

where $\lambda \geq 0$ is the regularization parameter that controls the level of penalization. The solution is

$$\hat{\boldsymbol{\beta}}_\lambda = \boldsymbol{\Phi}^\intercal(\boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal + n\lambda\mathbf{I})^{-1}\bar{\mathbf{y}}, \tag{16}$$

where $\mathbf{I}$ is the n-by-n identity matrix. The predictor for f is then

$$\hat{f}(\mathbf{x}) = \hat{\boldsymbol{\beta}}_\lambda^\intercal \boldsymbol{\phi}(\mathbf{x}) = (\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}))^\intercal(\boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal + n\lambda\mathbf{I})^{-1}\bar{\mathbf{y}}. \tag{17}$$

Note that

$$\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \begin{pmatrix} \boldsymbol{\phi}(\mathbf{x})^\intercal \boldsymbol{\phi}(\mathbf{x}_1) \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x})^\intercal \boldsymbol{\phi}(\mathbf{x}_n) \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal = \begin{pmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^\intercal \boldsymbol{\phi}(\mathbf{x}_1) & \cdots & \boldsymbol{\phi}(\mathbf{x}_1)^\intercal \boldsymbol{\phi}(\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi}(\mathbf{x}_n)^\intercal \boldsymbol{\phi}(\mathbf{x}_1) & \cdots & \boldsymbol{\phi}(\mathbf{x}_n)^\intercal \boldsymbol{\phi}(\mathbf{x}_n) \end{pmatrix}.$$

Thus, the predictor (17) depends on the basis functions $\boldsymbol{\phi}$ only through the product of the form $\boldsymbol{\phi}(\mathbf{x})^\intercal \boldsymbol{\phi}(\mathbf{x}')$ for some $\mathbf{x}$ and $\mathbf{x}'$. If we define a bivariate function $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\intercal \boldsymbol{\phi}(\mathbf{x}')$, then the predictor (17) can be written as

$$\hat{f}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\intercal (\mathbf{K} + n\lambda \mathbf{I})^{-1} \bar{\mathbf{y}}, \tag{18}$$

where $\mathbf{k}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n))^\intercal$ and $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_{i'}))_{i,i'=1}^n$. Hence, the predictor (17) is formally identical to the GP regression predictor (13), provided that $\mu \equiv 0$ and $\boldsymbol{\Sigma} = n\lambda \mathbf{I}$, i.e., $\sigma^2(\mathbf{x}_i)/r_i = n\lambda$ for all $i = 1, \ldots, n$.

This connection implies that the RLS method on linear basis function models may be interpreted, from a Bayesian perspective, as GP regression. That is, we impose on f a GP prior with mean 0 and covariance function $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\intercal \boldsymbol{\phi}(\mathbf{x}')$; moreover, the observation noise $\bar{\epsilon}_i$ is Gaussian with variance $n\lambda$. In particular, with $\lambda = 0$, we obtain an equivalence between the predictor (7) and the noise-free GP regression, which is also known as GP interpolation.

The converse way of interpretation—GP regression as RLS—is also available, but it involves the theory of reproducing kernel Hilbert spaces, which is beyond the scope of this tutorial. We refer interested readers to Kanagawa et al. (2018).

2.5. Enhancing Surrogates with Auxiliary Information

So far we have treated the simulation model as a black box that performs nothing but transforming inputs to (noisy) outputs. By doing so, we have implicitly assumed that the only data available when constructing a surrogate are simulation outputs. However, there possibly exists auxiliary information in practice that we can leverage, without much extra computational overhead, to enhance the prediction capability of the surrogate. We present below two general approaches.

2.5.1. Enhancement with Stylized Models Simulation models, by design, are used to describe in detail the interactions between the components that constitute a complex stochastic system. However, the main features of the same system may be captured by a stylized model that yields analytical expressions for the performance measure of interest as a function of the design variables, provided that sufficiently many simplifying assumptions are made.

Consider the following example in Shen et al. (2018). The patient flow through various medical units of a hospital is a complicated queueing network. The finite capacity of one medical unit to accommodate patients often results in blocking patients from entering, making them stay in

the upstream medical units even if the service has been completed there, and possibly creating further blocking. A stylized model—however crude it may appear—is to decompose the network into isolated independent units by discarding the interactions and model each unit as a multi-server queue. Performance measures such as mean length of stay can be derived in closed form for the stylized model.

Stylized models are often used to gain insights into the inner workings of the stochastic system of interest. But they can be easily incorporated to construct a surrogate for the simulation model. Let $\psi(\mathbf{x})$ denote the analytical expression derived from a stylized model. Then, we may simply add ψ to the set of generic basis functions and use the augmented set either directly in linear basis function models or to construct the mean function for GPs.

The key idea here is that one may use a crude—but computationally cheap otherwise—model to capture the overall trend of the response surface. Conceivably, the residual after de-trending will have fewer variations, thereby easier to fit by a surrogate. Thus, the requirement for ψ to possess an analytical expression can be relaxed as long as it can be computed sufficiently fast, e.g., from a low-fidelity simulation model. See Lin et al. (2019) for more discussion.

2.5.2. Enhancement with Gradient Observations Given simulation outputs for an input value $\mathbf{x}$, an estimate of the gradient of the response surface with respect to $\mathbf{x}$ can often be obtained with a negligible additional computational burden. This kind of *direct* gradient estimation—in contrast to finite-difference approximations—requires no re-simulation and can be achieved via infinitesimal perturbation analysis or the likelihood ratio/score function method (L'Ecuyer 1990) in many simulation applications, including queueing systems (Fu 2015) and financial engineering (Glasserman 2003, Chapter 7).

Suppose that in addition to $y_{i,\ell}$, the observation of $f(\mathbf{x}_i)$ for replication ℓ at point $\mathbf{x}_i$, we obtain an unbiased estimate of $\frac{\partial f(\mathbf{x}_i)}{\partial x_j}$, the partial derivative of $f(\mathbf{x}_i)$ with respect to the j-th coordinate, and denote it by $g_{i,j,\ell}$.

We first consider enhancing the linear basis function model with gradient observations. We present below a formulation that generalizes the approach in Fu and Qu (2014), which focuses on linear regression models. By doing so, it can be unified with the approach in Chen et al. (2013) that enhances GP surrogates with gradient observations.

Assume $f(\mathbf{x}) = \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{x})$ is differentiable. Then, the gradient surface is $\nabla f(\mathbf{x}) = \boldsymbol{\beta}^\intercal \nabla \boldsymbol{\phi}(\mathbf{x})$. Further, assume that for all $i = 1, \ldots, n$ and $\ell = 1, \ldots, r_i$,

$$\begin{aligned} y_{i,\ell} &= \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x}_i) + \epsilon_{i,\ell}, \\ g_{i,j,\ell} &= \boldsymbol{\beta}^\intercal \frac{\partial \boldsymbol{\phi}(\mathbf{x}_i)}{\partial x_j} + \zeta_{i,j,\ell}, \quad j = 1, \ldots, d, \end{aligned} \tag{19}$$

where the vector of noise terms $(\epsilon_{i,\ell}, \zeta_{i,1,\ell}, \ldots, \zeta_{i,d,\ell})$ has a multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix $\mathbf{V}$, but no dependence exists across i or ℓ. The correlation between, say, $\epsilon_{i,\ell}$ and $\zeta_{i,j,\ell}$ stems from the fact that the gradient estimate $g_{i,j,\ell}$ is usually computed based on $y_{i,\ell}$. Moreover, we do not consider the use of common random numbers here, which would introduce dependence across different design points.

Taking averages across replications in (19) results in

$$\begin{aligned} \bar{y}_i &= \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x}_i) + \bar{\epsilon}_i, \\ \bar{g}_{i,j} &= \boldsymbol{\beta}^\intercal \frac{\partial \boldsymbol{\phi}(\mathbf{x}_i)}{\partial x_j} + \bar{\zeta}_{i,j}, \quad j = 1, \ldots, d, \end{aligned} \tag{20}$$

where $\bar{g}_{i,j} \coloneqq {r_i}^{-1} \sum_{\ell=1}^{r_i} g_{i,j,\ell}$ and $\bar{\zeta}_{i,j} \coloneqq {r_i}^{-1} \sum_{\ell=1}^{r_i} \zeta_{i,j,\ell}$. Then, the system of Equations (20) can be viewed as a linear basis function model, with an augmented set of basis functions $\{\boldsymbol{\phi}(\mathbf{x}), \frac{\partial \boldsymbol{\phi}(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial \boldsymbol{\phi}(\mathbf{x})}{\partial x_d}\}$ and a vector of outputs $(\bar{y}_i, \bar{g}_{i,1}, \ldots, \bar{g}_{i,d})$. Due to the existence of correlations between the noise terms $\bar{\epsilon}_i, \bar{\zeta}_{i,1}, \ldots, \bar{\zeta}_{i,d}$, the OLS estimator of $\boldsymbol{\beta}$ is not statistically efficient; that is, there exists another estimator with a smaller variance. Instead, it is recommended to use the method of generalized least squares (GLS):

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{\mathsf{GLS}} &= \arg\min_{\boldsymbol{\beta}} (\bar{\mathbf{y}}_+ - \boldsymbol{\Phi}_+ \boldsymbol{\beta})^\intercal \mathbf{V}^{-1} (\bar{\mathbf{y}}_+ - \boldsymbol{\Phi}_+) \\ &= (\boldsymbol{\Phi}_+^\intercal \mathbf{V}^{-1} \boldsymbol{\Phi}_+)^{-1} \boldsymbol{\Phi}_+^\intercal \mathbf{V}^{-1} \bar{\mathbf{y}}_+, \end{aligned}$$

where

$$\bar{\mathbf{y}}_+ = \begin{pmatrix} \bar{y}_1 \\ \bar{g}_{1,1} \\ \vdots \\ \bar{g}_{1,d} \\ \vdots \\ \bar{y}_n \\ \bar{g}_{n,1} \\ \vdots \\ \bar{g}_{n,d} \end{pmatrix} \in \mathbb{R}^{n(d+1)} \quad \text{and} \quad \bar{\boldsymbol{\Phi}}_+ = \begin{pmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^\intercal \\ \left(\frac{\partial \boldsymbol{\phi}(\mathbf{x}_1)}{\partial x_1}\right)^\intercal \\ \vdots \\ \left(\frac{\partial \boldsymbol{\phi}(\mathbf{x}_1)}{\partial x_d}\right)^\intercal \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_n)^\intercal \\ \left(\frac{\partial \boldsymbol{\phi}(\mathbf{x}_n)}{\partial x_1}\right)^\intercal \\ \vdots \\ \left(\frac{\partial \boldsymbol{\phi}(\mathbf{x}_n)}{\partial x_d}\right)^\intercal \end{pmatrix} \in \mathbb{R}^{n(d+1) \times p}.$$

GP surrogates can also be enhanced with gradient observations. We briefly overview the approach proposed in Chen et al. (2013), and refer to Qu and Fu (2014) and Huo et al. (2018) for further developments.

Assume $f(\mathbf{x}) = \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x}) + \mathcal{M}(\mathbf{x})$, where $\mathcal{M}(\mathbf{x})$ is a zero-mean GP with covariance function $K(\mathbf{x}, \mathbf{x}')$. Then, the observations of the response surface and its gradient satisfy

$$\begin{aligned} \bar{y}_i &= \boldsymbol{\beta}^\intercal \boldsymbol{\phi}(\mathbf{x}_i) + \mathcal{M}(\mathbf{x}_i) + \bar{\epsilon}_i, \\ \bar{g}_{i,j} &= \boldsymbol{\beta}^\intercal \frac{\partial \boldsymbol{\phi}(\mathbf{x}_i)}{\partial x_j} + \frac{\partial \mathcal{M}(\mathbf{x}_i)}{\partial x_j} + \bar{\zeta}_{i,j}, \quad j = 1, \ldots, d, \end{aligned} \tag{21}$$

where the partial derivative of a GP $\frac{\partial \mathcal{M}(\mathbf{x})}{\partial x_j}$ is defined in a mean-square sense. Under regularity conditions, it can be shown that $(\mathcal{M}(\mathbf{x}), \frac{\partial \mathcal{M}(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial \mathcal{M}(\mathbf{x})}{\partial x_d})$ forms a multi-output GP with mean zero, and its covariance function can be derived explicitly by taking partial derivatives of $K(\mathbf{x}, \mathbf{x}')$. The prediction can be made in closed form, but the formula is fairly involved so we omit the details.

3. SO with Surrogates as Local Approximations

Based on convergence guarantees, SO algorithms may be classified into three categories: locally convergent algorithms that converge to the set of local optimal solutions or stationary points, globally convergent algorithms that converge to the set of global optimal solutions, and heuristic algorithms that have no convergence guarantee. Although global convergence is ideal, it is a global property, i.e., it typically requires exploring the entire feasible region in the limit. We will introduce many such algorithms in Section 4.

In many practical situations, however, the computational budget is limited and only allows exploring a small proportion of the feasible region. In these situations, local convergence that only requires local information becomes more meaningful, and it may be practically tested with a certain statistical guarantee; see, for instance, Bettonvil et al. (2009) and Xu et al. (2010). One way to obtain the local information about a solution is through a local surrogate, which allows the SO algorithm to check whether the solution is a local optimal solution and, if not, identify a direction (or a region) where better solutions may be found.

For SO purposes, due to Taylor's expansion, the most natural choices of local surrogates are low-order polynomials (see Section 2.1), especially first- and second-order polynomials. Response-surface methodology (RSM) is a collection of statistical methods that build on this idea to solve stochastic optimization problems, which include SO problems. We review RSM in Section 3.1. However, RSM often requires human involvement because it typically deals with real experiments (such as agricultural or clinical experiments), so it is not particularly suited for SO problems. The stochastic trust-region response-surface method (STRONG) of Chang et al. (2013) solves this problem by combing RSM with the trust-region method developed for deterministic nonlinear optimization. We review the STRONG and the related algorithms in Section 3.2. Other than low-order polynomials, other types of local surrogates have also been used in SO. We introduce the surrogate-based promising area search algorithm of Fan and Hu (2018) in Section 3.3.

3.1. Response Surface Methodology

RSM was first developed by Box and Wilson (1951) to optimize the operating conditions of a chemical process. It has evolved into a major tool for optimizing real (i.e., non-simulation) experiments. According to Myers et al. (2016), "RSM is a collection of statistical and mathematical techniques useful for developing, improving, and optimizing processes." It typically includes two stages. In the

first stage, it runs a number of experiments in a local region of the current solution and builds a first-order surrogate in the form of

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j. \tag{22}$$

Notice that Equation (22) implies that $\nabla f(\mathbf{x}) = (\beta_1, \ldots, \beta_d)^\intercal$. The surrogate essentially provides an ascent direction to allow RSM to find a better solution. Once it has a new solution, it repeats the process to find a better solution iteratively until the first-order model is no longer adequate. Then, RSM switches to the second stage, where it builds a second-order surrogate in the form of Equation (3) to locate the optimal solution.

Because RSM is typically used for expensive real experiments (or simulation experiments that are slow to run), large-sample properties, e.g., convergence or rate of convergence, are typically not considered in the literature, and the focus is mainly on statistical issues, such as the design of experiments (DOE) to efficiently estimate the surrogates and the tests of model inadequacy and optimality (Myers et al. 2016). For instance, there are $d+1$ and $d(d+1)/2+1$ parameters in the first- and second-order models, respectively. Different DOE schemes are proposed to reduce the required number of experiments to appropriately estimate these parameters (Kleijnen 2015).

RSM has also become a popular heuristic tool for SO (Hood and Welch 1993, Kleijnen 2008), especially when the simulation experiments are time-consuming. Much has been developed to understand how simulation experiments impact the statistical properties of RSM. For instance, Schruben and Margolin (1978) study how the use of common random numbers impacts the fitting of polynomials; Angün et al. (2009) consider stochastic constraints; and Bettonvil et al. (2009) develop tests of the Karush–Kuhn–Tucker conditions.

3.2. Stochastic Trust-Region Response-Surface Method

While RSM is a popular tool for SO, it has several problems, especially when simulation experiments are relatively fast (so that a large number of experiments may be conducted) and simulation noise is significant. First, it requires human involvement. For instance, in each iteration of the RSM, a surrogate needs to be optimized in a local region, but the local regions are determined by experimenters based on their experience. Moreover, the transitions between first- and second-order models typically also depend on human experience, and it is not clear whether a second-order model may transition back to a first-order model if it is found inadequate due to the simulation noise. Second, it is not clear whether the RSM algorithms have any convergence guarantees. This is a relevant question, especially when the number of simulation experiments becomes large.

Chang et al. (2013) propose the STRONG algorithm, which combines the trust-region method of deterministic nonlinear optimization with the RSM framework, to solve the two problems of RSM.

In any iteration, say iteration k, let $\mathbf{x}_k$ and Δ_k denote the current solution and the size of the trust region. STRONG conducts the following four steps:

Step 1. Construct a local model $r_k(\mathbf{x})$ around the current solution $\mathbf{x}$. If $\Delta_k \geq \tilde{\Delta}$, where $\tilde{\Delta}$ is a threshold, $r_k(\mathbf{x})$ is a first-order model; otherwise, $r_k(\mathbf{x})$ is a second-order model;

Step 2. Solve $\mathbf{x}_k^* \in \arg\max\{r_k(\mathbf{x}) : \mathbf{x} \in \mathcal{B}(\mathbf{x}_k, \Delta_k)\}$, where $\mathcal{B}(\mathbf{x}_k, \Delta_k)$ denotes a d-dimensional ball centered at $\mathbf{x}_k$ with a radius Δ_k, which is the trust region at iteration k;

Step 3. Simulate a number of observations at $\mathbf{x}_k^*$ and estimate $f(\mathbf{x}_k^*)$;

Step 4. Conduct the sufficient-reduction and ratio-comparison tests to update $\mathbf{x}_{k+1}$ and Δ_k.

In the algorithm, the sufficient-reduction test conducts new simulation experiments to test whether $\mathbf{x}_k^*$ is statistically significantly better than $\mathbf{x}_k$. If it is not, then the current solution is not updated, i.e., $\mathbf{x}_{k+1} = \mathbf{x}_k$, and the trust region shrinks, i.e., $\Delta_{k+1} = \gamma_1 \Delta_k$ where $0 < \gamma_1 < 1$ is a constant. If $\mathbf{x}_k^*$ passes the sufficient-reduction test, then the algorithm moves to the ratio-comparison test, which computes

$$\rho_k = \frac{\bar{f}_k(\mathbf{x}_k^*) - \bar{f}_k(\mathbf{x}_k)}{r_k(\mathbf{x}_k^*) - r_k(\mathbf{x}_k)},$$

where $\bar{f}_k(\mathbf{x}_k^*)$ and $\bar{f}_k(\mathbf{x}_k)$ are the estimated objective values (using simulation experiments) at $\mathbf{x}_k^*$ and $\mathbf{x}_k$, respectively. Notice that ρ_k denotes the ratio between the actual observed improvement and the predicted improvement. Let $0 < \eta_0 < \eta_1 < 1$ be two thresholds. If $\rho_k \geq \eta_1$, which implies that the local model works well, the algorithm then moves the current solution to the new solution, i.e., $\mathbf{x}_{k+1} = \mathbf{x}_k^*$, and enlarges the size of the trust region, i.e., $\Delta_{k+1} = \gamma_2 \Delta_k$ where $\gamma_2 > 1$ is a constant. If $\eta_0 \leq \rho_k < \eta_1$, which implies that the local model has some predictive power, the algorithm updates the new solution i.e., $\mathbf{x}_{k+1} = \mathbf{x}_k^*$, but keeps the size of the trust region, i.e., $\Delta_{k+1} = \Delta_k$. If $\rho_k < \eta_1$, which implies that the local model works poorly, the algorithm keeps the current solution, i.e., $\mathbf{x}_{k+1} = \mathbf{x}_k$, and shrinks the size of the trust region, i.e., $\Delta_{k+1} = \gamma_1 \Delta_k$.

Notice that the STRONG algorithm uses the trust region as the local region and its size Δ_k is adaptively updated based on the sufficient-reduction and ratio-comparison tests. Furthermore, the transitions between first- and second-order models are based on the size of the trust region Δ_k. If it is larger than the threshold $\tilde{\Delta}$, a first-order model is used; otherwise, a second-order model is used. This transition rule allows the algorithm to transition between the two models in both directions. Therefore, the algorithm gets rid of the human involvement that is necessary for typical RSM algorithms. Furthermore, Chang et al. (2013) show that, under certainly technical conditions on the estimated surrogates, the STRONG algorithm converges to a stationary point of the original SO problem.

The use of the trust-region method in SO has also been studied by others. For instance, Deng and Ferris (2009) combine it with the sample-average approximation to solve SO problems; Shashaani

et al. (2018) integrate it into a derivative-free algorithm to solve SO problems; and Mathesen et al. (2017) use it with a restart approach to design a globally convergent SO algorithm. The STRONG algorithm has also been extended by Chang et al. (2014) to include a screening stage so that it may solve large-scale SO problems with hundreds of dimensions.

3.3. Surrogate-Based Promising Area Search

In Sections 3.1 and 3.2 we introduced the RSM and STRONG algorithms. Both of them build low-order polynomials as local surrogates and use these models to guide the optimization process. In this subsection we introduce the Surrogate-Based Promising Area Search (SPAS) algorithm of Fan and Hu (2018) that allows the use of any interpolation surrogates, e.g., kriging, splines or radial basis functions.

Unlike RSM and STRONG, SPAS fits a global surrogate in each iteration. However, as Fan and Hu (2018) pointed out, "the use of the surrogate in our approach [i.e., SPAS] is not intended to provide a global fit of the underlying response surface, but rather aims to accurately predict the objective function values at unsampled points within the current search area." That's why we also include SPAS in Section 3, which focuses on SO algorithms with local surrogate approximations.

In each iteration, SPAS consists of the following four steps:

Step 1. Construct the most promising area (MPA), and sample a set of candidate solutions from it;

Step 2. Estimate the objective values of all visited solutions using a shrinking-ball method, which averages the samples in a d-dimensional ball centered at the solution;

Step 3. Build a surrogate that interpolates all these solutions;

Step 4. Find the optimal solution of the surrogate within the MPA.

Besides the use of surrogates, SPAS also integrates several other critical ideas of SO. The shrinking-ball method of estimating the function value at any solution was first introduced by Baumert and Smith (2002), and it has been studied and applied by Andradóttir and Prudius (2010) and Kiatsupaibul et al. (2018). The concept of the MPA was first proposed by Hong and Nelson (2006) in their COMPASS algorithm, which solves DOvS problems. Hong and Nelson (2007) further extend the idea into a general DOvS framework. By combining surrogate modeling, the shrinking-ball method and the MPA, SPAS is proved to converge to the set of local optimal solutions if the objective function is Lipschitz continuous.

4. SO with Surrogates as Global Approximations

There are mainly two strategies for using global surrogates such as linear basis function models and GPs to solve a continuous SO problem, depending on whether the design points $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ are chosen in a *static* fashion or a *sequential* one.

The former means that $\mathcal{X}$ is determined—once and for all—prior to any simulation experiments. With no observations of the response surface being available, a primary design principle of $\mathcal{X}$ is to cover the design space as much as possible so that most part of the response surface can be recovered after the observations are obtained. Typical experimental designs include lattice designs and space-filling designs, and we refer to Santner et al. (2003, Chapter 5) for details. Given $\mathcal{X}$, one runs simulation at each design point, possibly multiple times, fits a surrogate with the observations, and then optimizes the predicted surface $\hat{f}(\mathbf{x})$ induced by the surrogate. Being a deterministic function, $\hat{f}(\mathbf{x})$ can be optimized with any numerical optimization algorithms (Nocedal and Wright 2006). We refer to Barton and Meckesheimer (2006) for more discussion on the use of global surrogates in conjunction with static experimental designs.

Recent advances in SO with surrogates as global approximations are dominated by sequential experimental designs—design points are determined one at a time after each new observation of the response surface is made. Each new design point is selected based on (i) the updated surrogate reflecting the previous observations, and (ii) certain criteria that balance *exploration* and *exploitation*. By exploration, we mean searching the part of the design space that has much uncertainty; by exploitation, we mean searching the area in the proximity of the current best solution. The need for quantifying the uncertainty about the response surface renders GPs the most popular class of surrogates. (Recall that posterior distributions of a GP can be derived in closed form and the computation is reduced to linear algebra.) This line of research is closely related to Bayesian optimization (Shahriari et al. 2016, Frazier 2018) in the machine learning literature, where a primary motivation is hyperparameter tuning of sophisticated machine learning algorithms/models (Feurer and Hutter 2019). We introduce below three representative examples of such methods: knowledge gradient (Scott et al. 2011), upper confidence bound (Srinivas et al. 2012), and GP-based search (Sun et al. 2014). All three methods share the following structure procedure-wise:

Step 1. Impose a GP prior on f;

Step 2. Select the next batch of design points subject to a prescribed "criterion" that is computed using the current belief about f;

Step 3. Run simulation experiments at each of the newly selected design points;

Step 4. Update the GP posterior given the new observations of f via Equations (11) and (12);

Step 5. Repeat Steps 2–4 until the simulation budget is exhausted;

Step 6. Optimize the posterior mean function and return the optimum as a solution to problem (1).

As demonstrated below, GP-based sequential methods for continuous SO problems mainly differ in how to define the criterion in Step 2 for selecting new design points.

4.1. Knowledge Gradient

Knowledge gradient (KG) was originally proposed to solve R&S problems (Frazier et al. 2008, 2009). The method was generalized in Scott et al. (2011) to cover continuous SO problems.

Suppose that simulation experiments have been made at $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with one replication each, generating observations $y_i = f(\mathbf{x}_i) + \epsilon_i$, $i = 1, \ldots, n$. We are interested in selecting the next design point $\mathbf{x}_{n+1}$. Let $\mathcal{D}_n = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$. Let μ_n and K_n be the posterior mean and covariance functions of f conditional on $\mathcal{D}$. Following Equations (11) and (12), it can be shown that μ_n and K_n satisfy the following updating scheme:

$$\mu_{n+1}(\mathbf{x}) = \mu_n(\mathbf{x}) + \delta_n(\mathbf{x}, \mathbf{x}_{n+1}) Z_{n+1}, \tag{23}$$

$$K_{n+1}(\mathbf{x}, \mathbf{x}') = K_n(\mathbf{x}, \mathbf{x}') - \delta_n(\mathbf{x}, \mathbf{x}_{n+1}) \delta_n(\mathbf{x}', \mathbf{x}_{n+1}), \tag{24}$$

where

$$Z_{n+1} \coloneqq \frac{y_{n+1} - \mu_n(\mathbf{x}_{n+1})}{\sqrt{K_n(\mathbf{x}_{n+1}, \mathbf{x}_{n+1}) + \sigma^2(\mathbf{x}_{n+1})}} \quad \text{and} \quad \delta_n(\mathbf{x}, \mathbf{v}) \coloneqq \frac{K_n(\mathbf{x}, \mathbf{v})}{\sqrt{K_n(\mathbf{v}, \mathbf{v}) + \sigma^2(\mathbf{v})}};$$

moreover, Z_{n+1} is a standard normal random variable conditional on $\mathcal{D}_n$.

The KG method selects $\mathbf{x}_{n+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} \mathrm{KG}_n(\mathbf{x})$, where

$$\mathrm{KG}_n(\mathbf{x}) \coloneqq \mathbb{E}\Big[\max_{\mathbf{u} \in \mathcal{X}} \mu_{n+1}(\mathbf{u}) - \max_{\mathbf{u} \in \mathcal{X}} \mu_n(\mathbf{u}) \Big| \mathcal{D}_n, \mathbf{x}_{n+1} = \mathbf{x} \Big]. \tag{25}$$

The interpretation of $\mathrm{KG}_n(\mathbf{x})$ is as follows. If our simulation budget were exhausted after collecting data $\mathcal{D}_n$, then we would use $\max_{\mathbf{u}} \mu_n(\mathbf{u})$ to estimate the maximum value of f. However, now that we are allowed to run one more simulation experiment at $\mathbf{x}_{n+1}$, the posterior mean function will become $\mu_{n+1}(\cdot)$ in Equation (23) after the new data point $(\mathbf{x}_{n+1}, y_{n+1})$ is obtained. Thus, the increment in the estimated maximum value of f is $\max_{\mathbf{u}} \mu_{n+1}(\mathbf{u}) - \max_{\mathbf{u}} \mu_n(\mathbf{u})$. Before the simulation is run at $\mathbf{x}_{n+1}$, this increment is a random variable conditional on $\mathcal{D}_n$, and its distribution is determined by the standard normal random variable Z_{n+1} in Equation (23).

There are several approaches for the numerical maximization of $\mathrm{KG}_n(\mathbf{x})$. Scott et al. (2011) propose maximizing

$$\widetilde{\mathrm{KG}}_n(\mathbf{x}) \coloneqq \mathbb{E}\Big[\max_{1 \le i \le n+1} \mu_{n+1}(\mathbf{x}_i) - \max_{1 \le i \le n+1} \mu_n(\mathbf{x}_i) \Big| \mathcal{D}_n, \mathbf{x}_{n+1} = \mathbf{x} \Big],$$

a discrete proxy of $\mathrm{KG}_n(\mathbf{x})$, because $\widetilde{\mathrm{KG}}_n(\mathbf{x})$ and its gradient with respect to $\mathbf{x}$ can both be computed explicitly.

A second approach to solving $\max_{\mathbf{x}} \mathrm{KG}_n(\mathbf{x})$ is to view it as a stochastic optimization problem, in which the only random variable involved is the standard normal Z_{n+1}. Then, one may apply sample average approximation (Kim et al. 2015) by simulating realizations of Z_{n+1}.

Yet another approach, proposed in Wu and Frazier (2016), is to apply stochastic approximation (Chau and Fu 2015). Note that, under mild regularity conditions,

$$\nabla_{\mathbf{x}}\mathrm{KG}_n(\mathbf{x}) = \nabla_{\mathbf{x}}\,\mathbb{E}\Big[\max_{\mathbf{u}\in\mathcal{X}}\mu_{n+1}(\mathbf{u}) - \max_{\mathbf{u}\in\mathcal{X}}\mu_n(\mathbf{u})\big|\mathcal{D}_n,\mathbf{x}_{n+1}=\mathbf{x}\Big] = \mathbb{E}\Big[\nabla_{\mathbf{x}}\max_{\mathbf{u}\in\mathcal{X}}\mu_{n+1}(\mathbf{u})\big|\mathcal{D}_n,\mathbf{x}_{n+1}=\mathbf{x}\Big].$$

Hence, $\nabla_{\mathbf{x}}\max_{\mathbf{u}\in\mathcal{X}}\mu_{n+1}(\mathbf{u})$ is an unbiased estimator of $\nabla_{\mathbf{x}}\mathrm{KG}_n(\mathbf{x})$, and it can be computed by applying the envelope theorem (Milgrom and Segal 2002).

4.2. Upper Confidence Bound

Upper confidence bound (UCB) is a celebrated class of methods for multi-armed bandit (MAB) problems. Similar to R&S problems, MAB problems are also concerned with finding the optimal among a finite set of alternatives with unknown performances/rewards. A key difference between the two classes of problems lies in the objective. Basically, MAB can be viewed as an online decision-making problem that aims to maximize the cumulative rewards collected over the entire time horizon; in contrast, R&S is more of an offline flavor and focuses on the final reward collected at the end of the time horizon. We refer to Slivkins (2019) and Auer (2002) for introductions to MAB problems and UCB-type algorithms, respectively.

Srinivas et al. (2012) generalize UCB to the setting of optimizing a GP sample path. The general structure of the GP-UCB method is basically the same as that of the KG method, except that the next design point is selected as $\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in\mathcal{X}}\mathrm{UCB}_n(\mathbf{x})$, where

$$\mathrm{UCB}_n(\mathbf{x}) \coloneqq \mu_n(\mathbf{x}) + \sqrt{\gamma_n K_n(\mathbf{x},\mathbf{x})}, \tag{26}$$

and $\gamma_n > 0$ is a tuning parameter that should grow as a function of n. The form of $\mathrm{UCB}_n(\mathbf{x})$ clearly shows the trade-off between exploration and exploitation. A potential design point $\mathbf{x}$ is favored if either $\mu_n(\mathbf{x})$ is large (exploitation), or $K_n(\mathbf{x},\mathbf{x})$ is large (exploration).

Evidently, $\mathrm{UCB}_n(\mathbf{x})$ is much easier to maximize than $\mathrm{KG}_n(\mathbf{x})$, as the former involves no expectation. Nevertheless, a potential downside of the GP-UCB method is that its performance depends critically on the choice of γ_n. It might be tempting to make γ_n grow at a rate of $\ln(n)$, which is both a typical choice for MAB problems (Auer 2002) and the choice analyzed in Srinivas et al. (2012). However, such a choice is recommended with the aim of maximizing the cumulative reward of the form $\sum_{i=1}^{n} f(\mathbf{x}_i)$. There may conceivably exist a better guideline for setting γ_n for problem (1). This issue is yet to be addressed.

4.3. GP-Based Search

Random search is a general class of algorithms for SO, which samples randomly a number of design points in each iteration of the algorithm based a sampling distribution that may be updated based on all information collected through the optimization process. Random search algorithms are most

popular for DOvS problems, but they have also been developed to solve continuous SO problems; see, for instance, the recent reviews of Hong et al. (2015) and Andradóttir (2015). One of the critical issues in developing random search algorithms is how to design a sampling distribution that automatically balances exploration and exploitation. As pointed out earlier in this section, GP is capable of quantifying the uncertainty of the response surface. Therefore, it can be used to facilitate the design of good sampling distributions, which is the basic idea of the GP-based Search (GPS) algorithm of Sun et al. (2014).

Sun et al. (2014) note that the posterior mean function $\mu_n(\mathbf{x})$ and the posterior variance function $\sigma_n^2(\mathbf{x}) = K_n(\mathbf{x}, \mathbf{x})$ of Equations (11) and (12) provide information on exploitation and exploration, respectively. In particular, higher mean values indicate the region needs more exploitation and higher variance values indicate the region needs more exploration. Therefore, the GPS algorithm uses the following sampling distribution

$$h(\mathbf{x}) = \frac{\Pr\{Z(\mathbf{x}) > c\}}{\sum_{\mathbf{z}\in\mathcal{X}} \Pr\{Z(\mathbf{z}) > c\}},$$

where $\mathcal{X}$ is a finite set of discrete solutions, c is set as the current estimated optimal value, and $Z(\mathbf{x})$ follows a normal distribution with mean $\mu_n(\mathbf{x})$ and variance $\sigma_n^2(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$. Notice that the sampling distribution combines both the mean and variance information and balances exploration and exploitation seamlessly.

To use the sampling distribution there are two remaining issues. The first is how to sample from the distribution. Notice that the denominator of $h(\mathbf{x})$ involves a summation that is typically difficult to compute. Sun et al. (2014) solve the problem by developing an acceptance-rejection algorithm and a Markov chain Monte Carlo algorithm to sample from the distribution.

The second issue is the calculation of $\mu_n(\mathbf{x})$ and $\sigma_n^2(\mathbf{x})$ using Equations (11) and (12). Notice that the calculation involves a matrix inversion. When the number of design points is large, this calculation is time-consuming. Furthermore, when the design points are close to each other (which is common in later iterations of the algorithm when it focuses more on exploiting the good regions), the matrix is often ill-conditioned and the inversion becomes difficult. To solve the problem, the GPS algorithm takes a very pragmatic view towards the GP. Instead of considering the objective value as a sample path from the GP, as in Bayesian optimization algorithms, the GPS algorithm only treats it as a surrogate approximation that facilitates the generation of good sampling distributions. It proposes the following GP to model the objective function:

$$f(\mathbf{x}) = \mathcal{M}(\mathbf{x}) + \lambda(\mathbf{x})^\intercal(\bar{\mathbf{y}} - \mathbf{M}) + \lambda(\mathbf{x})^\intercal\mathcal{E}, \tag{27}$$

where $\mathcal{M}(\mathbf{x})$ is an unconditional GP, $\lambda(\mathbf{x}) = (\lambda_1(\mathbf{x}), \ldots, \lambda_n(\mathbf{x}))^\intercal$ is a vector of weight functions, $\mathbf{M} = (M(\mathbf{x}_1), \ldots, M(\mathbf{x}_n))^\intercal$ is a vector of $\mathcal{M}(\mathbf{x})$ evaluated at $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and $\mathcal{E} = (\epsilon_1, \ldots, \epsilon_n)^\intercal$ is an

n-dimensional random vector following a multivariate normal distribution with the mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$, which is the n-by-n diagonal matrix with the i-th diagonal element being $\sigma^2(\mathbf{x}_i)/r_i$. Furthermore, in Equation (27), $\mathcal{M}(\mathbf{x})$ and $\mathcal{E}$ are independent of each other and $\bar{\mathbf{y}}$ is considered deterministic when building the model.

Let $\tilde{\mu}_n(\mathbf{x})$ and $\tilde{\sigma}_n^2(\mathbf{x})$ denote the mean and variance functions of the new GP model. When the weight function vector $\lambda(\mathbf{x})$ is continuous in $\mathbf{x}$ and it satisfies $\lambda_i(\mathbf{x}) \geq 0$, $\sum_{i=1}^n \lambda_i(\mathbf{x}) = 1$ and $\lambda_i(\mathbf{x}_j) = 1\{\mathbf{x}_i = \mathbf{x}_j\}$, where $1\{\cdot\}$ is the indicator function, Sun et al. (2014) show that

$$\begin{aligned}\tilde{\mu}_n(\mathbf{x}) &= \lambda(\mathbf{x})^\intercal \bar{\mathbf{y}},\\ \tilde{\sigma}_n^2(\mathbf{x}) &= K(\mathbf{x},\mathbf{x}) - 2\lambda(\mathbf{x})^\intercal \mathbf{k}(\mathbf{x}) + \lambda(\mathbf{x})^\intercal(\mathbf{K}+\boldsymbol{\Sigma})\lambda(\mathbf{x}).\end{aligned}$$

Then, both $\tilde{\mu}_n(\mathbf{x})$ and $\tilde{\sigma}_n^2(\mathbf{x})$ may be calculated directly without matrix inversion. Furthermore, Sun et al. (2014) show that $\tilde{\mu}_n(\mathbf{x})$ interpolates all design points through their sample means, i.e., $\tilde{\mu}_n(\mathbf{x}_i) = \bar{y}_i$ for all $i = 1,\ldots,n$, and $\tilde{\mu}_n(\mathbf{x})$ and $\tilde{\sigma}_n^2(\mathbf{x})$ capture the information of exploitation and exploration and, therefore, can be used to construct the sampling distribution in the GPS algorithm.

The GPS algorithm has global convergence (Sun et al. 2014). However, it is designed to solve DOvS problems. Sun et al. (2018) extend it to solve continuous SO problems and proved its global convergence. To further simplify the sampling process, Sun et al. (2018) propose to approximate the sampling distribution by a Gaussian mixture model that can be sampled easily, and show that the resulting algorithm is still globally convergent.

5. Computation for Large Datasets

We have taken it for granted thus far in this tutorial that surrogates are computationally fast, and rightly so in light of their analytical tractability. However, this postulate is challenged when the number of design points is large because it usually involves numerically inverting a large matrix to process the simulation data $\{(\mathbf{x}_i, \bar{y}_i) : i = 1,\ldots,n\}$; see, e.g., Equation (7) for linear basis function models, as well as Equations (11) and (12) for GPs. It is well known that the time complexity for matrix inversion scales as $\mathcal{O}(n^3)$ in general. As n grows, surrogates are increasingly demanding in computation and eventually become even more expensive than the simulation model that they aim to approximate in the first place (Huang et al. 2006, Sun et al. 2014, Salemi et al. 2019b).

The need for processing large datasets calls for approximation methods to reduce the computational burden caused by numerical inversion of large matrices. There exists a huge literature on approximate computation for GP regression; see Liu et al. (2020) for a recent survey. We present two popular methods—the Nyström method and random features. Both methods have drawn substantial interest in recent years.

5.1. The Nyström Method

A central idea to mitigate the challenge of computing $(\mathbf{K}+\mathbf{\Sigma})^{-1}$ is to find a low-rank approximation of $\mathbf{K}$. In particular, consider a rank-m matrix of the form $\tilde{\mathbf{K}} = \mathbf{UCV}$, where $\mathbf{U} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{m\times m}$, and $\mathbf{V} \in \mathbb{R}^{m\times n}$ with $m < n$. If $\mathbf{C}$ is invertible, the Woodbury matrix identity (Horn and Johnson 2012, page 19) asserts that

$$(\mathbf{K}+\mathbf{\Sigma})^{-1} \approx (\tilde{\mathbf{K}}+\mathbf{\Sigma})^{-1} = \mathbf{\Sigma}^{-1} - \mathbf{\Sigma}^{-1}\mathbf{U}(\mathbf{C}^{-1}+\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{\Sigma}^{-1}. \tag{28}$$

Then, the computational bottleneck has been transformed to the inversion of smaller, m-by-m matrices. (The inversion of $\mathbf{\Sigma}$ is easy because it is a diagonal matrix.)

For ease of presentation, let $I = \{1,\dots,n\}$ and $A \subset I$ be a subset of size m, which is also called the active set of indices. The Nyström method was originally devised to approximate the eigenfunctions of a covariance function $K(\cdot,\cdot)$; see, e.g., Williams and Seeger (2001). It suggests the following low-rank approximation of the covariance matrix $\mathbf{K}$:

$$\tilde{\mathbf{K}} = \mathbf{K}_{n,m}\mathbf{K}_{m,m}^{-1}\mathbf{K}_{m,n}, \tag{29}$$

where $\mathbf{K}_{n,m} \coloneqq (K(\mathbf{x}_i,\mathbf{x}_{i'}))_{i\in I,i'\in A}$, $\mathbf{K}_{m,m} \coloneqq (K(\mathbf{x}_i,\mathbf{x}_{i'}))_{i\in A,i'\in A}$, and $\mathbf{K}_{m,n} \coloneqq (K(\mathbf{x}_i,\mathbf{x}_{i'}))_{i\in A,i'\in I}$. Then, the posterior mean function in Equation (11) can be approximated by replacing $\mathbf{K}$ with $\tilde{\mathbf{K}}$:

$$\begin{aligned}
&\mathbb{E}[f(\mathbf{x})|\mathcal{D}_n] \\
&\approx \mu(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\intercal(\tilde{\mathbf{K}}+\mathbf{\Sigma})^{-1}(\bar{\mathbf{y}}-\boldsymbol{\mu}) \\
&= \mu(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\intercal\mathbf{\Sigma}^{-1}(\bar{\mathbf{y}}-\boldsymbol{\mu}) - \mathbf{k}(\mathbf{x})^\intercal\mathbf{\Sigma}^{-1}\mathbf{K}_{n,m}(\mathbf{K}_{m,n}+\mathbf{K}_{m,n}\mathbf{\Sigma}^{-1}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\mathbf{\Sigma}^{-1}(\bar{\mathbf{y}}-\boldsymbol{\mu}),
\end{aligned} \tag{30}$$

where the last step follows from (28). Despite its long expression, the approximation (30) can be computed in time $\mathcal{O}(m^2n)$. In addition, the posterior covariance function in Equation (12) can be approximated in a similar fashion:

$$\begin{aligned}
&\mathrm{Cov}[f(\mathbf{x}),f(\mathbf{x}')|\mathcal{D}_n] \\
&\approx K(\mathbf{x},\mathbf{x}') - \mathbf{k}(\mathbf{x})^\intercal(\tilde{\mathbf{K}}+\mathbf{\Sigma})^{-1}\mathbf{k}(\mathbf{x}') \\
&= K(\mathbf{x},\mathbf{x}') - \mathbf{k}(\mathbf{x})^\intercal\mathbf{\Sigma}^{-1}\mathbf{k}(\mathbf{x}') + \mathbf{k}(\mathbf{x})^\intercal\mathbf{\Sigma}^{-1}\mathbf{K}_{n,m}(\mathbf{K}_{m,n}+\mathbf{K}_{m,n}\mathbf{\Sigma}^{-1}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\mathbf{\Sigma}^{-1}\mathbf{k}(\mathbf{x}'),
\end{aligned} \tag{31}$$

whose time complexity is also $\mathcal{O}(m^2n)$.

However, there is a caveat in the above use of the low-rank approximation (29). That is, using (31) to approximate the posterior variance $\mathrm{Var}[f(\mathbf{x})|\mathcal{D}_n] = \mathrm{Cov}[f(\mathbf{x}),f(\mathbf{x})|\mathcal{D}_n]$ may yield a negative value! A better implementation of the Nyström method is to construct a covariance function $\tilde{K}(\cdot,\cdot)$ to systematically replace the occurrences of $K(\cdot,\cdot)$.

Specifically, define $\tilde{K}(\mathbf{x},\mathbf{x}') \coloneqq \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{m,m}^{-1} \mathbf{k}_m(\mathbf{x}')$, where $\mathbf{k}_m(\mathbf{x}) \in \mathbb{R}^m$ is the vector composed of $K(\mathbf{x},\mathbf{x}_i)$ for all $i \in A$. It is easy to show that (i) $\tilde{K}$ is a covariance function, and (ii) the covariance matrix associated with evaluating $\tilde{K}$ at $\{\mathbf{x}_1,\dots,\mathbf{x}_n\}$ is identical to $\tilde{\mathbf{K}}$ in Equation (29). Let $\tilde{f}$ denote a GP with mean function μ and covariance function $\tilde{K}$. Then, we can use the posterior distribution of $\tilde{f}$ to approximate that of f. This treatment is adopted by Smola and Schölkopf (2000) and Rudi et al. (2015); see also Lu et al. (2020) for recent advances. It contrasts the approximations (30) and (31) which simply replace the occurrences of $\mathbf{K}$ in the posterior distribution of f with $\tilde{\mathbf{K}}$.

In particular, it can be shown (see Appendix A) that

$$\mathbb{E}[\tilde{f}(\mathbf{x})|\mathcal{D}_n] = \mu(\mathbf{x}) + \mathbf{k}_m(\mathbf{x})^\intercal (\mathbf{K}_{m,m} + \mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}(\bar{\mathbf{y}} - \boldsymbol{\mu}), \tag{32}$$

$$\mathrm{Cov}[\tilde{f}(\mathbf{x}),\tilde{f}(\mathbf{x}')|\mathcal{D}_n] = K(\mathbf{x},\mathbf{x}') - \mathbf{k}_m(\mathbf{x})^\intercal (\mathbf{K}_{m,m} + \mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{n,m})^{-1}\mathbf{k}_m(\mathbf{x}'), \tag{33}$$

both of which can be computed with time complexity $\mathcal{O}(m^2 n)$. We stress, nonetheless, that (32)/(33) are not identical to (30)/(31).

At last, we briefly comment on choosing m and A. Although theoretical analysis may relate the accuracy of the approximation with the asymptotic order of magnitude of m relative to n, in practice m is usually viewed as a tuning parameter. One may gradually increase the value of m, evaluate the resulting accuracy of the approximation, and stop when the marginal improvement falls below some prescribed threshold; see more discussion in Lu et al. (2020). Given m, A may be determined by random sampling from the entire set of indices $\{1,\dots,n\}$.

5.2. Random Features

Random features represents a large class of algorithms for approximating covariance functions (Liu et al. 2021). We introduce below the original version, called random Fourier features (RFF). It is proposed in Rahimi and Recht (2007)—the seminal work that gives rise to this research direction.

Similar to the Nyström method, RFF also seeks to construct another covariance function $\tilde{K}$ that yields a low-rank approximation to the covariance matrix $\mathbf{K}$ to accelerate computation. However, the approximation that RFF constructs is *data-independent*; that is, it does not depend on the design points $\{\mathbf{x}_1,\dots,\mathbf{x}_n\}$. This is in contrast to the Nyström method, for which the approximations are defined through the design points in the active set.

RFF applies particularly to *stationary* covariance functions, including both the Gaussian and Matérn classes. If K is stationary covariance function, then Bochner's theorem (Stein 1999, page 24) asserts that it can be represented as the Fourier transform of a non-negative finite measure:

$$K(\mathbf{x},\mathbf{x}') = K(\mathbf{0},\mathbf{0}) \int_{\mathbb{R}^d} e^{\mathrm{i}\boldsymbol{\omega}^\intercal(\mathbf{x}-\mathbf{x}')} \mathrm{p}(d\boldsymbol{\omega}), \tag{34}$$

where $\mathsf{p}(\cdot)$ is a probability measure on $\mathbb{R}^d$. For example, if K is the Gaussian covariance function in Equation (8), then $\mathsf{p}(\cdot)$ corresponds to the multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix $\eta^{-2}\mathbf{I}$. More examples of the (K, p) pair can be found in Liu et al. (2021).

Because $K(\mathbf{x}, \mathbf{x}')$ is real-valued, we may discard the imaginary part on the right-hand-side of Equation (34). Thus,

$$K(\mathbf{x}, \mathbf{x}') = K(\mathbf{0}, \mathbf{0}) \int_{\mathbb{R}^d} \cos\left(\boldsymbol{\omega}^\intercal(\mathbf{x} - \mathbf{x}')\right) \mathsf{p}(d\boldsymbol{\omega}).$$

Further, if $\boldsymbol{\omega} \in \mathbb{R}^d$ is a random vector having distribution p, then

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= K(\mathbf{0}, \mathbf{0})\, \mathbb{E}_{\boldsymbol{\omega}} \left[\cos\left(\boldsymbol{\omega}^\intercal(\mathbf{x} - \mathbf{x}')\right)\right] \\ &= K(\mathbf{0}, \mathbf{0})\, \mathbb{E}_{\boldsymbol{\omega}, b} \left[\sqrt{2}\cos\left(\boldsymbol{\omega}^\intercal \mathbf{x} + b\right) \sqrt{2}\cos\left(\boldsymbol{\omega}^\intercal \mathbf{x}' + b\right)\right], \end{aligned} \tag{35}$$

where b is an independent random variable uniformly distributed on $(0, 2\pi)$. The proof of Equation (35) is provided in Appendix B.

We then apply the standard Monte Carlo approximation:

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &\approx K(\mathbf{0}, \mathbf{0}) \cdot \frac{1}{m} \sum_{t=1}^{m} \sqrt{2}\cos\left(\boldsymbol{\omega}_t^\intercal \mathbf{x} + b_t\right) \sqrt{2}\cos\left(\boldsymbol{\omega}_t^\intercal \mathbf{x}' + b_t\right) \\ &= \left(\sqrt{\tfrac{2K(\mathbf{0},\mathbf{0})}{m}}\cos\left(\boldsymbol{\omega}_1^\intercal \mathbf{x} + b_1\right) \cdots \sqrt{\tfrac{2K(\mathbf{0},\mathbf{0})}{m}}\cos\left(\boldsymbol{\omega}_m^\intercal \mathbf{x} + b_m\right)\right) \begin{pmatrix} \sqrt{\frac{2K(\mathbf{0},\mathbf{0})}{m}}\cos\left(\boldsymbol{\omega}_1^\intercal \mathbf{x}' + b_1\right) \\ \vdots \\ \sqrt{\frac{2K(\mathbf{0},\mathbf{0})}{m}}\cos\left(\boldsymbol{\omega}_m^\intercal \mathbf{x}' + b_m\right) \end{pmatrix} \\ &\coloneqq \boldsymbol{\phi}_m(\mathbf{x})^\intercal \boldsymbol{\phi}_m(\mathbf{x}') \coloneqq \tilde{K}(\mathbf{x}, \mathbf{x}'), \end{aligned}$$

where $\{\omega_t : t = 1, \ldots, m\}$ are independent samples drawn from p, and $\{b_t : t = 1, \ldots, m\}$ are independent samples drawn from $\mathsf{Uniform}(0, 2\pi)$. It is easy to show that $\tilde{K}$ is a covariance function.

Note that, in light of the discussion in Section 2.4, $\boldsymbol{\phi}_m(\mathbf{x})$ can be view as a vector of basis functions—also known as *features* in the machine learning literature—and they are constructed via random sampling, hence the method's name "random features."

Let $\tilde{\mathbf{K}} \coloneqq (\tilde{K}(\mathbf{x}_i, \mathbf{x}_{i'}))_{i,i'=1}^n$. Then,

$$\tilde{\mathbf{K}} = \begin{pmatrix} \boldsymbol{\phi}_m(\mathbf{x}_1)^\intercal \\ \vdots \\ \boldsymbol{\phi}_m(\mathbf{x}_n)^\intercal \end{pmatrix} \left(\boldsymbol{\phi}_m(\mathbf{x}_1) \cdots \boldsymbol{\phi}_m(\mathbf{x}_n)\right) \coloneqq \boldsymbol{\Phi}_m \boldsymbol{\Phi}_m^\intercal.$$

Because $\boldsymbol{\Phi}_m$ is a n-by-m matrix, $\tilde{\mathbf{K}}$ is a low-rank approximation of the covariance matrix $\mathbf{K}$, provided that $m < n$. In addition, let $\tilde{f} \sim \mathsf{GP}(\mu, \tilde{K})$. We prove in Appendix B that

$$\mathbb{E}[\tilde{f}(\mathbf{x}) | \mathcal{D}_n] = \mu(\mathbf{x}) + \boldsymbol{\phi}_m(\mathbf{x})^\intercal (\mathbf{I} + \boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi}_m)^{-1} \boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1} (\bar{\mathbf{y}} - \boldsymbol{\mu}), \tag{36}$$

$$\mathrm{Cov}[\tilde{f}(\mathbf{x}), \tilde{f}(\mathbf{x}') | \mathcal{D}_n] = K(\mathbf{x}, \mathbf{x}') - \boldsymbol{\phi}_m(\mathbf{x})^\intercal (\mathbf{I} + \boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi}_m)^{-1} \boldsymbol{\phi}_m(\mathbf{x}'). \tag{37}$$

Similar to the Nyström approximations (32)–(33), the time complexity for computing (36)–(37) is also $\mathcal{O}(m^2 n)$.

6. Concluding Remarks

We have introduced several common surrogates—low-order polynomials, linear basis function models, and Gaussian processes—along with simple techniques to enhance their prediction capability with little additional computational overhead. With the use of these surrogates, we have presented a number of approaches to solving SO problems with continuous decision variables, some of which (RSM, STRONG, SPAS) are locally convergent while others (KG, GP-UCB, GPS) globally convergent. Moreover, we have discussed two widely popular methods—the Nyström method and random features—for dealing with computational challenges associated with Gaussian processes in the presence of large datasets.

Looking forward, we believe the following research directions are potentially fruitful and of high impact. First, an ideal SO algorithm in our mind would be able to quickly identify most local optima and then select the best among them. In contrast, it is often observed in practice that globally convergent methods such as those introduced in this tutorial tend to "over-explore"—direct sampling efforts away from the proximity of a local optimum without realizing it. Lack of gradient information to provide local curvature of the response surface, among others, is an important reason. It is of great interest to develop algorithms that integrate local search and global search to ensure fast convergence to global optima.

Second, another plausible approach to accelerating convergence to global optima is to incorporate structural information—such as monotonicity, convexity, and level of differentiability—provided that one can safely impose such assumptions on the response surface. Surrogates that process such properties do exist (Lim and Glynn 2012, Wang and Berger 2016, Salemi et al. 2019a). But how to leverage them to develop fast SO algorithms is yet to be fully investigated. See Zhang and Zheng (2020) and Zhang et al. (2020) for recent developments in this regard.

Third, most theoretical analyses of SO algorithms in the literature focus on proving convergence to a local/global optimum as the computational budget grows. Such asymptotic analysis can hardly explain the algorithms' performance in finite time, nor can it provide accurate guidance for performance tuning in practice. Little is known about their rates of convergence. Notable exceptions include Bull (2011) and Bouttier and Gavra (2019). The former establishes the rate of convergence for Efficient Global Optimization algorithms (Jones et al. 1998), while the latter for Simulated Annealing algorithms (Gelfand and Mitter 1989). In general, deeper theoretical understanding of SO algorithms is strongly needed to fill the gap, which may shed light on critical attributes required for improving algorithm efficiency.

Fourth, in Section 3.2 we introduced the STRONG algorithm that integrates low-order polynomial surrogates with trust-region methods. Even though there are a number of papers taking this approach, we think it is under-studied and has a potential to solve large-scale SO problems. To build

a full quadratic model, one needs at least $d(d+1)/2+1$ design points and each design point may need more than one replication of the simulation experiments in order to take into consideration the simulation noise. If one needs to conduct these many experiments in each iteration, it may be too costly. One way that may solve the problem is to use L_1-regularization, a.k.a. LASSO, in the regression to select the most important parameters of the quadratic model with far fewer experiments (Tibshirani 1996). Notice that these selected parameters include an important part of the ascent and curvature information and one may use this partial information to guide the optimization process. Even though we think the idea of adding L_1-regularization into the trust-region framework has great potential for SO, it is quite challenging to design efficient algorithms and to analyze their asymptotic properties, e.g., convergence and rate of convergence.

Fifth, running simulation experiments is often time-consuming. But different simulation experiments are typically independent and they can be run on different processors. Therefore, it is natural to think how to design SO algorithms that work well in parallel computing environments. Recently, Luo et al. (2015), Ni et al. (2017), Zhong and Hong (2021), and others have developed parallel algorithms for R&S problems. Wu and Frazier (2016) also develop parallel knowledge gradient algorithms for Bayesian optimization. However, we believe there are still many opportunities in developing efficient surrogate-based parallel SO algorithms.

Last but not the least, in the present era of big data an emerging decision-making paradigm that has gained great popularity in recent years is optimization with covariates. That is, the optimal decision is no longer constant, but varies as a function of the covariates that represent the additional contextual information available at the moment of making a decision; see, e.g., Ban and Rudin (2019), Bertsimas and Kallus (2020), and Bertsimas and Koduri (2021). However, these articles focus on settings where the objective functions are analytically tractable. Shen et al. (2021) address the problem of R&S with covariates, assuming the response surface for each alternative is a linear function in the covariates. We expect much more to be explored in the direction of SO with covariates for years to come.

Appendix A: Proofs Related to the Nyström Method

Let $\tilde{K}(\mathbf{x},\mathbf{x}') \coloneqq \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{m,m}^{-1}\mathbf{k}_m(\mathbf{x}')$, $\tilde{\mathbf{k}}(\mathbf{x}) \coloneqq (\tilde{K}(\mathbf{x},\mathbf{x}_1),\ldots,\tilde{K}(\mathbf{x},\mathbf{x}_n))^\intercal$, and $\mathbf{Q} \coloneqq \mathbf{K}_{m,m}+\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{n,m}$. Then,

$$
\begin{aligned}
\tilde{\mathbf{k}}(\mathbf{x})^\intercal(\tilde{\mathbf{K}}+\boldsymbol{\Sigma})^{-1} &= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{m,m}^{-1}\mathbf{K}_{m,n}(\mathbf{K}_{n,m}\mathbf{K}_{m,m}^{-1}\mathbf{K}_{m,n}+\boldsymbol{\Sigma})^{-1} \\
&= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{m,m}^{-1}\mathbf{K}_{m,n}[\boldsymbol{\Sigma}^{-1}-\boldsymbol{\Sigma}^{-1}\mathbf{K}_{n,m}(\mathbf{K}_{m,m}+\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{n,m})^{-1}\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}] \qquad (38)\\
&= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{m,m}^{-1}(\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}-\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{n,m}\mathbf{Q}^{-1}\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}) \\
&= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{m,m}^{-1}(\mathbf{I}-\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{n,m}\mathbf{Q}^{-1})\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1} \\
&= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{m,m}^{-1}(\mathbf{Q}-\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{n,m})\mathbf{Q}^{-1}\mathbf{K}_{m,n}\boldsymbol{\Sigma}^{-1}
\end{aligned}
$$

$$= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{Q}^{-1} \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1}, \tag{39}$$

where Equation (38) follows from the Woodbury matrix identity.

Applying Equation (11) to $\tilde{f} \sim \mathsf{GP}(\mu, \tilde{K})$, we have

$$\begin{aligned}
\mathbb{E}[\tilde{f}(\mathbf{x})|\mathcal{D}_n] &= \mu(\mathbf{x}) + \tilde{\mathbf{k}}(\mathbf{x})^\intercal (\tilde{\mathbf{K}} + \boldsymbol{\Sigma})^{-1} (\bar{\mathbf{y}} - \boldsymbol{\mu}) \\
&= \mu(\mathbf{x}) + \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{Q}^{-1} \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} (\bar{\mathbf{y}} - \boldsymbol{\mu}) \\
&= \mu(\mathbf{x}) + \mathbf{k}_m(\mathbf{x})^\intercal (\mathbf{K}_{m,m} + \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{n,m})^{-1} \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} (\bar{\mathbf{y}} - \boldsymbol{\mu}),
\end{aligned}$$

where the second equation follows from (39). This completes the proof of Equation (32).

Also by Equation (39),

$$\begin{aligned}
\tilde{\mathbf{k}}(\mathbf{x})^\intercal (\tilde{\mathbf{K}} + \boldsymbol{\Sigma})^{-1} \tilde{\mathbf{k}}(\mathbf{x}') &= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{Q}^{-1} \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{k}}(\mathbf{x}') \\
&= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{Q}^{-1} \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{n,m} \mathbf{K}_{m,m}^{-1} \mathbf{k}_m(\mathbf{x}').
\end{aligned}$$

It follows that, after applying Equation (12) to $\tilde{f} \sim \mathsf{GP}(\mu, \tilde{K})$,

$$\begin{aligned}
K(\mathbf{x}, \mathbf{x}') - \mathrm{Cov}[\tilde{f}(\mathbf{x}), \tilde{f}(\mathbf{x}')|\mathcal{D}_n] &= \tilde{\mathbf{k}}(\mathbf{x})^\intercal (\tilde{\mathbf{K}} + \boldsymbol{\Sigma})^{-1} \tilde{\mathbf{k}}(\mathbf{x}') \\
&= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{m,m}^{-1} \mathbf{k}_m(\mathbf{x}) - \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{Q}^{-1} \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{n,m} \mathbf{K}_{m,m}^{-1} \mathbf{k}_m(\mathbf{x}') \\
&= \mathbf{k}_m(\mathbf{x})^\intercal (\mathbf{I} - \mathbf{Q}^{-1} \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{n,m}) \mathbf{K}_{m,m}^{-1} \mathbf{k}_m(\mathbf{x}') \\
&= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{Q}^{-1} (\mathbf{Q} - \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{n,m}) \mathbf{K}_{m,m}^{-1} \mathbf{k}_m(\mathbf{x}') \\
&= \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{Q}^{-1} \mathbf{k}_m(\mathbf{x}') \\
&= \mathbf{k}_m(\mathbf{x})^\intercal (\mathbf{K}_{m,m} + \mathbf{K}_{m,n} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{n,m})^{-1} \mathbf{k}_m(\mathbf{x}'),
\end{aligned}$$

proving Equation (33).

Appendix B: Proofs Related to Random Features

Suppose $b \sim \mathsf{Uniform}(0, 2\pi)$. Then, for any $a \in \mathbb{R}$,

$$\mathbb{E}_b[\cos(a + 2b)] = \int_0^{2\pi} \frac{\cos(a+2b)}{2\pi}\, db = \frac{1}{2\pi} \sin(a+2b)\Big|_0^{2\pi} = \frac{1}{2\pi}[\sin(a + 4\pi) - \sin(a)] = 0.$$

Thus,

$$\mathbb{E}_{\boldsymbol{\omega},b}\big[\cos\big(\boldsymbol{\omega}^\intercal(\mathbf{x} + \mathbf{x}') + 2b\big)\big] = \mathbb{E}_{\boldsymbol{\omega}}\big[\mathbb{E}_b\big[\cos\big(\boldsymbol{\omega}^\intercal(\mathbf{x} + \mathbf{x}') + 2b\big)\big|\boldsymbol{\omega}\big]\big] = 0.$$

It follows that

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\omega},b}\big[\cos\big(\boldsymbol{\omega}^\intercal(\mathbf{x} - \mathbf{x}')\big)\big] &= \mathbb{E}_{\boldsymbol{\omega},b}\big[\cos\big(\boldsymbol{\omega}^\intercal(\mathbf{x} + \mathbf{x}') + 2b\big)\big] + \mathbb{E}_{\boldsymbol{\omega},b}\big[\cos\big(\boldsymbol{\omega}^\intercal(\mathbf{x} - \mathbf{x}')\big)\big] \\
&= \mathbb{E}_{\boldsymbol{\omega},b}\big[\cos\big((\boldsymbol{\omega}^\intercal\mathbf{x} + 2b) + (\boldsymbol{\omega}^\intercal\mathbf{x}' + 2b)\big)\big] + \mathbb{E}_{\boldsymbol{\omega},b}\big[\cos\big((\boldsymbol{\omega}^\intercal\mathbf{x} + 2b) - (\boldsymbol{\omega}^\intercal\mathbf{x}' + 2b)\big)\big] \\
&= \mathbb{E}_{\boldsymbol{\omega},b}\big[2\cos(\boldsymbol{\omega}^\intercal\mathbf{x} + b)\cos(\boldsymbol{\omega}^\intercal\mathbf{x}' + b)\big],
\end{aligned}$$

proving Equation (35).

We now prove Equations (36) and (37), following the strategy in Appendix A. Let $\tilde{K}(\mathbf{x}, \mathbf{x}') \coloneqq \boldsymbol{\phi}_m(\mathbf{x})^\intercal \boldsymbol{\phi}_m(\mathbf{x}')$ and $\mathbf{Q} \coloneqq \mathbf{I} + \boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi}_m$. The key is to derive $\tilde{\mathbf{k}}(\mathbf{x})^\intercal (\tilde{\mathbf{K}} + \boldsymbol{\Sigma})^{-1}$ as follows.

$$\tilde{\mathbf{k}}(\mathbf{x})^\intercal (\tilde{\mathbf{K}} + \boldsymbol{\Sigma})^{-1} = \boldsymbol{\phi}_m(\mathbf{x})^\intercal \boldsymbol{\Phi}_m^\intercal (\boldsymbol{\Phi}_m \boldsymbol{\Phi}_m^\intercal + \boldsymbol{\Sigma})^{-1}$$

$$
\begin{aligned}
&= \boldsymbol{\phi}_m(\mathbf{x})^\intercal \boldsymbol{\Phi}_m^\intercal [\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}_m(\mathbf{I} + \boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}_m)^{-1}\boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1}] \\
&= \boldsymbol{\phi}_m(\mathbf{x})^\intercal (\boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1} - \boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}_m \mathbf{Q}^{-1}\boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1}) \\
&= \boldsymbol{\phi}_m(\mathbf{x})^\intercal (\mathbf{I} - \boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}_m \mathbf{Q}^{-1})\boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1} \\
&= \boldsymbol{\phi}_m(\mathbf{x})^\intercal (\mathbf{Q} - \boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}_m)\mathbf{Q}^{-1}\boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1} \\
&= \boldsymbol{\phi}_m(\mathbf{x})^\intercal \mathbf{Q}^{-1}\boldsymbol{\Phi}_m^\intercal \boldsymbol{\Sigma}^{-1}.
\end{aligned}
$$

The remaining calculations are essentially the same as those in Appendix A, so we omit them.

Acknowledgments

L. Jeff Hong is supported in part by the Natural Science Foundation of China (Project No. 72091211 and Project No. 71991473). Xiaowei Zhang is supported in part by the Hong Kong Research Grant Council (Project No. 16211417 and Project No. 17201520).

References

Andradóttir S (2015) A review of random search methods. Fu MC, ed., *Handbook of Simulation Optimization*, 277–292 (New York: Springer).

Andradóttir S, Prudius AA (2010) Adaptive random search for continuous simulation optimization. *Naval Research Logistics* 57(6):583–604.

Angün E, Kleijnen J, den Hertog D, Gürkan G (2009) Response surface methodology with stochastic constraints for expensive simulation. *Journal of the Operational Research Society* 60(6):735–746.

Ankenman B, Nelson BL, Staum J (2010) Stochastic kriging for simulation metamodeling. *Operations Research* 58(2):371–382.

Auer P (2002) Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3:397–422.

Ban GY, Rudin C (2019) The big data newsvendor: Practical insights from machine learning. *Operations Research* 67(1):90–108.

Barton RR (2009) Simulation optimization using metamodels. Rossetti MD, Hill RR, Johansson B, Dunkin A, Ingalls RG, eds., *Proceedings of the 2009 Winter Simulation Conference*, 230–238 (Piscataway, NJ: IEEE).

Barton RR, Meckesheimer M (2006) Metamodel-based simulation optimization. Henderson S, Nelson B, eds., *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, chapter 18, 535–574 (Elsevier).

Baumert S, Smith RL (2002) Pure random search for noisy objective functions. Technical report, University of Michigan.

Bertsimas D, Kallus N (2020) From predictive to prescriptive analytics. *Management Science* 66(3):1025–1044.

Bertsimas D, Koduri N (2021) Data-driven optimization: A reproducing kernel Hilbert space approach. *Operations Research*, forthcoming, URL `https://doi.org/10.1287/opre.2020.2069`.

Bettonvil B, del Castillo E, Kleijnen JP (2009) Statistical testing of optimality conditions in multiresponse simulation-based optimization. *European Journal of Operational Research* 199(2):448–458.

Bishop C (2006) *Pattern Recognition and Machine Learning* (New York: Springer).

Bouttier C, Gavra I (2019) Convergence rate of a simulated annealing algorithm with noisy observations. *Journal of Machine Learning Research* 20(4):1–45.

Box GEP, Wilson KB (1951) On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)* 13(1):1–45.

Bull AD (2011) Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research* 12:2879–2904.

Chang KH, Hong LJ, Wan H (2013) Stochastic trust-region response-surface method (STRONG)—A new response-surface framework for simulation optimization. *INFORMS Journal on Computing* 25(2):230–243.

Chang KH, Li MK, Wan H (2014) Combining STRONG with screening designs for large-scale simulation optimization. *IIE Transactions* 46(4):357–373.

Chau M, Fu MC (2015) An overview of stochastic approximation. Fu MC, ed., *Handbook of Simulation Optimization*, 149–178 (New York: Springer).

Chen X, Ankenman B, Nelson BL (2013) Enhancing stochastic kriging metamodels with gradient estimators. *Operations Research* 61(2):512–528.

Deng G, Ferris MC (2009) Variable-number sample-path optimization. *Mathematical Programming* 117(1):81–109.

Fan Q, Hu J (2018) Surrogate-based promising area search for Lipschitz continuous simulation optimization. *INFORMS Journal on Computing* 30(4):677–693.

Feurer M, Hutter F (2019) Hyperparameter optimization. Hutter F, Kotthoff L, Vanschoren J, eds., *Automated Machine Learning: Methods, Systems, Challenges*, 3–33 (Cham, Switzerland: Springer Nature Switzerland AG).

Frazier P, Powell W, Dayanik S (2009) The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing* 21(4):599–613.

Frazier PI (2018) Bayesian optimization. Gel E, Ntaimo L, eds., *Recent Advances in Optimization and Modeling of Contemporary Problems*, 255–278, INFORMS TutORials in Operations Research (INFORMS).

Frazier PI, Powell W, Dayanik S (2008) A knowledge gradient policy for sequential information collection. *SIAM Journal on Control and Optimization* 47(5):2410–2439.

Fu MC (2015) Stochastic gradient estimation. Fu MC, ed., *Handbook of Simulation Optimization*, 105–147 (New York: Springer).

Fu MC, Qu H (2014) Regression models augmented with direct stochastic gradient estimators. *INFORMS Journal on Computing* 26(3):484–499.

Gelfand SB, Mitter SK (1989) Simulated annealing with noisy or imprecise energy measurements. *Journal of Optimization Theory and Applications* 62(1):49–62.

Glasserman P (2003) *Monte Carlo Methods in Financial Engineering* (New York: Springer).

Hastie T (2020) Ridge regularization: An essential concept in data science. *Technometrics* 62(4):426–433.

Hastie T, Tibshirani R, Friedman J (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (New York: Springer), 2nd edition.

Hong LJ, Nelson BL (2006) Discrete optimization via simulation using COMPASS. *Operations Research* 54(1):115–129.

Hong LJ, Nelson BL (2007) A framework for locally convergent random-search algorithms for discrete optimization via simulation. *ACM Transactions on Modeling and Computer Simulation* 17(4):Article 19.

Hong LJ, Nelson BL, Xu J (2015) Discrete optimization via simulation. Fu MC, ed., *Handbook of Simulation Optimization*, 9–44 (New York: Springer).

Hood SJ, Welch PD (1993) Response surface methodology and its application in simulation. Evans GW, Mollaghasemi M, Russell EC, Biles W, eds., *Proceedings of the 1993 Winter Simulation Conference*, 115–122 (Piscataway, NJ: IEEE).

Horn RA, Johnson CR (2012) *Matrix Analysis* (Cambridge University Press), 2nd edition.

Huang D, Allen TT, Notz WI, Zeng N (2006) Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization* 34(3):441–466.

Huo H, Zhang X, Zheng Z (2018) A scalable approach to enhancing stochastic kriging with gradients. Rabe M, Juan AA, Mustafee N, Skoogh A, Jain S, Johansson B, eds., *Proceedings of the 2018 Winter Simulation Conference*, 2213–2224 (Piscataway, NJ: IEEE).

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13(4):455–492.

Kanagawa M, Hennig P, Sejdinovic D, Sriperumbudur BK (2018) Gaussian processes and kernel methods: A review on connections and equivalences. Preprint available at *arXiv:1805.08845*.

Kiatsupaibul S, Smith RL, Zabinsky ZB (2018) Single observation adaptive search for continuous simulation optimization. *Operations Research* 66(6):1713–1727.

Kim S, Pasupathy R, Henderson SG (2015) A guide to sample average approximation. Fu MC, ed., *Handbook of Simulation Optimization*, 207–243 (New York: Springer).

Kleijnen JP (2008) Response surface methodology for constrained simulation optimization: An overview. *Simulation Modelling Practice and Theory* 16(1):50–64.

Kleijnen JPC (2015) Response surface methodology. Fu MC, ed., *Handbook of Simulation Optimization*, 81–104 (New York: Springer).

Krige DG (1951) *A statistical approach to some mine valuations and allied problems at the Witwatersrand*. Master's thesis, University of Witwatersrand, Johannesburg, South Africa.

L'Ecuyer P (1990) A unified view of the IPA, SF, and LR gradient estimation techniques. *Management Science* 36(11):1364–1383.

Lim E, Glynn PW (2012) Consistency of multidimensional convex regression. *Operations Research* 60(1):196–208.

Lin Z, Matta A, Shanthikumar JG (2019) Combining simulation experiments and analytical models with area-based accuracy for performance evaluation of manufacturing systems. *IISE Transactions* 51(3):266–283.

Liu F, Huang X, Chen Y, Suykens JA (2021) Random features for kernel approximation: A survey on algorithms, theory, and beyond. Preprint available at *arXiv:2004.11154*.

Liu H, Ong YS, Shen X, Cai J (2020) When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems* 31(11):4405–4423.

Lu X, Rudi A, Borgonovo E, Rosasco L (2020) Faster kriging: Facing high-dimensional simulators. *Operations Research* 68(1):233–249.

Luo J, Hong LJ, Nelson BL, Wu Y (2015) Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. *Operations Research* 63(5):1177–1194.

Matheron G (1963) Principles of geostatistics. *Econ. Geol.* 58(8):1246–1266.

Mathesen L, Pedrielli G, Ng SH (2017) Trust region based stochastic optimization with adaptive restart: A family of global optimization algorithms. Chan WKV, D'Ambrogio A, Zacharewicz G, Mustafee N, Wainer G, Page E, eds., *Proceedings of the 2017 Winter Simulation Conference*, 2104–2115 (Piscataway, NJ: IEEE).

Milgrom P, Segal I (2002) Envelope theorems for arbitrary choice sets. *Econometrica* 70(2):583–601.

Myers RH, Montgomery DC, Anderson-Cook CM (2016) *Response Surface Methodology: Process and Product Optimization Using Designed Experiments* (Wiley), 4th edition.

Ni EC, Ciocan DF, Henderson SG, Hunter SR (2017) Efficient ranking and selection in parallel computing environments. *Operations Research* 65(3):821–836.

Nocedal J, Wright S (2006) *Numerical Optimization* (New York: Springer), 2nd edition.

Qu H, Fu MC (2014) Gradient extrapolated stochastic kriging. *ACM Transactions on Modeling and Computer Simulation* 24(4):Article 23.

Rahimi A, Recht B (2007) Random features for large-scale kernel machines. Platt J, Koller D, Singer Y, Roweis S, eds., *Advances in Neural Information Processing Systems*, volume 20, 1177–1184 (Red Hook, NY: Curran Associates, Inc.).

Rasmussen CE, Williams KI (2006) *Gaussian Processes for Machine Learning* (MIT Press).

Rice JA (2007) *Mathematical Statistics and Data Analysis* (Duxbury), 3rd edition.

Rudi A, Camoriano R, Rosasco L (2015) Less is more: Nyström computational regularization. Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R, eds., *Advances in Neural Information Processing Systems*, volume 28, 1657–1665 (Red Hook, NY: Curran Associates, Inc.).

Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989) Design and analysis of computer experiments. *Statistical Science* 4(4):409–435.

Salemi P, Staum J, Nelson BL (2019a) Generalized integrated Brownian fields for simulation metamodeling. *Operations Research* 67(3):874–891.

Salemi PL, Song E, Nelson BL, Staum J (2019b) Gaussian Markov random fields for discrete optimization via simulation: Framework and algorithms. *Operations Research* 67(1):250–266.

Santner TJ, Williams BJ, Notz WI (2003) *The Design and Analysis of Computer Experiments* (New York: Springer).

Schruben LW, Margolin BH (1978) Pseudorandom number assignment in statistically designed simulation and distribution sampling experiments. *Journal of the American Statistical Association* 73(363):504–520.

Scott W, Frazier P, Powell W (2011) The correlated knowledge gradient for simulation optimization of continuous parameters using Gaussian process regression. *SIAM Journal on Optimization* 21(3):996–1026.

Shahriari B, Swersky K, Wang Z, Adams RP, de Freitas N (2016) Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE* 104(1):148–175.

Shashaani S, Hashemi FS, Pasupathy R (2018) ASTRO-DF: A class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization. *SIAM Journal on Optimization* 28(4):3145–3176.

Shen H, Hong LJ, Zhang X (2018) Stochastic kriging for queueing simulation with stylized models. *IISE Transactions* 50(11):943–958.

Shen H, Hong LJ, Zhang X (2021) Ranking and selection with covariates for personalized decision making. *INFORMS Journal on Computing*, forthcoming, URL `https://doi.org/10.1287/ijoc.2020.1009`.

Slivkins A (2019) Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* 12(1-2):1–286.

Smola AJ, Schölkopf B (2000) Sparse greedy matrix approximation for mahine learning. Langley P, ed., *Proceedings of the 17th International Conference on Machine Learning*, 911–918 (San Francisco, CA: Morgan Kaufmann Publishers).

Srinivas N, Krause A, Kakade SM, Seeger MW (2012) Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory* 58(5):3250–3265.

Stein ML (1999) *Interpolation of Spatial Data: Some Theory for Kriging* (New York: Springer).

Sun L, Hong LJ, Hu Z (2014) Balancing exploitation and exploration in discrete optimization via simulation through a Gaussian process-based search. *Operations Research* 62(6):1416–1438.

Sun W, Hu Z, Hong LJ (2018) Gaussian mixture model-based random search for continuous optimization via simulation. Rabe M, Juan AA, Mustafee N, Skoogh A, Jain S, Johansson B, eds., *Proceedings of the 2018 Winter Simulation Conference*, 2003–2014 (Piscataway, NJ: IEEE).

Tibshirani R (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 58(1):267–288.

Wang X, Berger JO (2016) Estimating shape constrained functions using Gaussian processes. *SIAM/ASA Journal on Uncertainty Quantification* 4(1):1–25.

Williams C, Seeger M (2001) Using the Nyström method to speed up kernel machines. Leen T, Dietterich T, Tresp V, eds., *Advances in Neural Information Processing Systems*, volume 13, 682–688 (MIT Press).

Wu J, Frazier PI (2016) The parallel knowledge gradient method for batch Bayesian optimization. Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R, eds., *Advances in Neural Information Processing Systems*, volume 29, 3126–3134 (Red Hook, NY: Curran Associates, Inc.).

Xu J, Nelson BL, Hong JL (2010) Industrial strength COMPASS: A comprehensive algorithm and software for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation* 20(1):Article 3.

Zhang H, Zheng Z (2020) Gradient-based algorithms for discrete convex optimization via simulation. Preprint available at *arXiv:2010.16250*.

Zhang H, Zheng Z, Lavaei J (2020) Stochastic localization methods for discrete convex optimization via simulation. Preprint available at *arXiv:2012.02427*.

Zhong Y, Hong LJ (2021) Knockout-tournament procedures for large-scale ranking and selection in parallel computing environments. *Operations Research*, forthcoming, URL `https://doi.org/10.1287/opre.2020.2065`.