Constrained Optimization in Random Simulation

Angun, Ebru; Kleijnen, Jack

Publication date:
2022

Document Version
Early version, also known as pre-print

No. 2022-022

CONSTRAINED OPTIMIZATION IN RANDOM SIMULATION: EFFICIENT GLOBAL OPTIMIZATION AND KARUSH-KUHN-TUCKER CONDITIONS

By

Ebru Angün, Jack P.C. Kleijnen

30 August 2022

ISSN 0924-7815
ISSN 2213-9532

Constrained optimization in random simulation: efficient global optimization and Karush-Kuhn-Tucker conditions

Ebru Angün (a) and Jack P.C. Kleijnen (b)

(a) Industrial Engineering Department

Galatasaray University

Istanbul, Turkey

E-mail: eangun@gsu.edu.tr

Orcid: https://orcid.org/ 0000-0002-8199-9746

(b) Department of Management

Tilburg School of Economics and Management

Tilburg University

Postbox 90153, 5000 LE Tilburg, Netherlands

E-mail: kleijnen@tilburguniversity.edu

Orcid: https://orcid.org/0000-0001-8413-2366

Abstract

We develop a novel method for solving constrained optimization problems in random (or stochastic) simulation; i.e., our method minimizes the goal output subject to one or more output constraints and input constraints. Our method is indeed novel, as it combines the Karush-Kuhn-Tucker (KKT) conditions with the popular algorithm called "efficient global optimization" (EGO), which is also known as "Bayesian optimization" and is related to "active learning". Originally, EGO solves non-constrained optimization problems in deterministic simulation; EGO is a sequential algorithm that uses Kriging (or Gaussian process) metamodeling of the underlying simulation model, treating the simulation as a black box. Though there are many variants of EGO—for these non-constrained deterministic problems and for variants of these problems—none of these EGO-variants use the KKT conditions—even though these conditions are well-known (first-order necessary) optimality conditions in white-box problems. Because the simulation is random, we apply stochastic Kriging. Furthermore, we allow for variance heterogeneity

and apply a popular sample allocation rule to determine the number of replicated simulation outputs for selected combinations of simulation inputs. Moreover, our algorithm can take advantage of parallel computing. We numerically compare the performance of our algorithm and the popular proprietary OptQuest algorithm, in two familiar examples (namely, a mathematical toy example and a practical inventory system with a service-level constraint); we conclude that our algorithm is more efficient (requires fewer expensive simulation runs) and effective (gives better estimates of the true global optimum).

Keywords: Simulation, design of experiments; simulation, statistical analysis; artificial intelligence; computational-experiments; inventory-production

JEL: C0, C1, C9, C15, C44

1 Introduction

In this paper we solve constrained optimization problems in *random* or *stochastic* simulation, combining *efficient global optimization* (EGO) and the *Karush-Kuhn-Tucker* (KKT or briefly KT) conditions; Kleijnen et al. (2022) investigates a similar solution, in the simpler context of *deterministic* simulation. EGO is a popular sequential statistical algorithm that uses *Kriging*—or *Gaussian process* (GP) metamodeling—and was originally developed for non-constrained optimization in deterministic simulation; see Jones et al. (1998). EGO is also known as *Bayesian optimization* (BO), and is related to *active learning*; see the many references in Frazier (2018). Though there are many variants of EGO (for non-constrained optimization problems and for variants of these problems), none of these EGO-variants use the KKT conditions—even though these conditions are well-known first-order necessary optimality conditions in white-box optimization; see Boyd and Vandenberghe (2009).

To denote the multiple random *simulation outputs*, we use the symbol w_h ($h = 0, 1, ..., t-1$) where $E(w_0)$ is the goal output to be minimized and $E(w_{h'})$ ($h' = 1, ..., t-1$) are the $(t-1)$ constrained outputs such that the output constraints are $E(w_{h'}) \leq c_{h'}$ with prespecified $c_{h'}$. For example, in inventory management, the users may wish to minimize the expected cost such that the expected out-of-stock fraction does not exceed 10%. Like Kleijnen et al. (2022), we assume k continuous *simulation inputs* x_j ($j = 1, ..., k$). Whereas Kleijnen et al. (2022) assumes that these x_j vary across a given *rectangular* experimental domain $l_j \leq x_j \leq u_j$, we assume— like Kleijnen et al. (2010)—more general input constraints; namely, $f_g(\boldsymbol{x}) \leq c_g$ ($g = 1, ...v$) such as $x_1 \leq x_2$. Furthermore, we assume that k is relatively small (say) $1 \leq k \leq 20$ (it is well-known that Kriging requires small k: "curse of dimensionality"). For example, $k = 2$ in inventory management of a single item (or stock keeping unit, SKU) through the popular (s, S) system (with $S > s$). We define the

k-dimensional vector $\boldsymbol{x} = (x_1, ..., x_k)'$.

Compared with deterministic simulation, our problem is more complicated because $E[w_h(\boldsymbol{x})]$ must be estimated. We estimate this $E[w_h(\boldsymbol{x})]$ from (say) m *replications*; by definition, replications give *independently and identically distributed* (IID) $w_{h;r}$ and $w_{h;r'}$ with $r \neq r'$ $(r, r' = 1, ..., m)$. So, replications use nonoverlapping (independent) streams of *pseudorandom numbers* (PRN). L'Ecuyer (2015) reviews PRN generators—in sequential and parallel computing—focussing on multiple independent streams and substreams, including software libraries (Obviously, the t output types are statistically dependent or correlated instead of IID.)

In our analysis we allow for *variance heterogeneity*; i.e., $var[w_h(\boldsymbol{x})]$ depends on $\boldsymbol{x}$, and denotes the so-called *intrinsic* variance caused by the PRN. Jalali et al. (2017) investigates variance heterogeneity in *un*constrained optimization solved through several variants of EGO. Ankenman et al. (2010) also investigates variance heterogeneity and derives *stochastic Kriging* (SK), but focuses on prediction (instead of optimization) so the SK performance is measured through the *mean squared prediction error* (MSPE).

Moreover, we focus on random simulation with *independent random numbers* (IRN) and $m(\boldsymbol{x}_i)$ replications where $\boldsymbol{x}_i$ denotes $\boldsymbol{x}$ in the simulated input combination i where $i = 1, ..., n$. We use the popular *sample allocation* rule $m(\boldsymbol{x}_i)/m(\boldsymbol{x}_{i'}) = var[w(\boldsymbol{x}_i)]/var[w(\boldsymbol{x}_{i'})$.

Note that *common random numbers* (CRN) with $m(\boldsymbol{x}_i) = m$ is usually analyzed by SK requiring $m > n$. We also derive *ordinary Kriging* (OK) with CRN requiring $m > 1$. CRN are meant to reduce the variance of the estimated difference between two means. In constrained optimization, however, we need to estimate the magnitude of the slacks $c_{h'} - w_{h'}(\boldsymbol{x})$; CRN may give a relatively bad estimator of these slacks.

The literature on Kriging for constrained optimization in random simulation is extremely limited. Wang and Ierapetritou (2018) considers a single output constraint, whereas we consider the general case of $t \geq 1$ constraints; moreover, that publication assumes IRN with $m_i = m$ and independent w_0 (objective) and w_1 (constraint), whereas our (s, S) inventory with a service-level (or fill-rate) constraint gives correlated outputs. Whereas EGO originally used either the *expected improvement* (EI) or the *probability of feasibility* (PI) infill criterion (or acquisition function), that publication proposes "feasibility-enhanced Expected Improvement" that combines the well-known *augmented expected improvement* (AEI) and PI; we use neither AEI, nor PI. A survey of optimization including random simulation is Mahévas et al. (2019). A recent publication using PI is Guidetti et al. (2022).

Our algorithm can take advantage of *parallel computing*; i.e., we implement our algorithm on a personal computer (PC) with 16 cores, and use MATLAB's default with 12 cores. Guidetti et al. (2022) develops a parallel BO procedure for constrained optimization, which uses PI. Note that Qing et al. (2022) develops a parallel procedure for constrained multi-objective optimization using a Pareto frontier and—instead of EI—an entropy infill criterion; Luo et al. (2015) develops a parallel procedure for so-called ranking-and-selection

(R&S); obviously, those problems are related to—yet different from—our type of problems.

We numerically investigate the performance of our algorithm, in a mathematical toy example inspired by Gramacy et al. (2016, p. 2) (and also used in Pourmohamad and Lee (2020) and Pourmohamad and Lee (2021)) and an (s, S) inventory system with a service-level constraint originally investigated in Bashyam and Fu (1998).

We organize the rest of this paper as follows. Section 2 formalizes our constrained optimization problem and the corresponding KT conditions; this section also interprets these conditions as a *least squares* (LS) model. Section 3 summarizes SK, derives SK estimators of the output gradients ∇_h, and discusses choosing the number of simulation replications m_i and input combinations n. Section 4 details a novel variant of EGO for optimization in random simulation with output and input constraints; i.e., this algorithm minimizes the well-known *modified EI* (or MEI) multiplied by a cosine that quantifies how well the KT conditions hold. Section 5 designs a *Monte Carlo* (MC) experiment with the toy example that controls the degree of variance heterogeneity; Subsection 5.1 compares the performance of our algorithm and OptQuest (which is a popular proprietary simulation optimization algorithm that combines several algorithms). Section 6 investigates an (s, S) inventory discrete-event simulation model with a service-level constraint (this constraint is used because practitioners find it hard to quantify out-of-stock costs, including loss-of-goodwill; this model is the major building block in practical inventory models with multiple SKUs); Subsection 6.1 compares the performance of our algorithm and OptQuest. Section 7 presents conclusions.

2 Constrained optimization and Kuhn-Tucker conditions

In Section 1, we mentioned that we focus on

$$\begin{aligned} &\min_{\boldsymbol{x}} E[w_0(\boldsymbol{x})] \text{ subject to} \\ &E[w_{h'}(\boldsymbol{x})] \leq c_{h'} \text{ with } h' = 1, \ldots, t-1, \\ &f_g(\boldsymbol{x}) \leq c_g \text{ with } g = 1, ..., v. \end{aligned} \tag{1}$$

Obviously, $f_g(\boldsymbol{x}) \leq c_g$ includes the usual *box constraints*

$$\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \ (\boldsymbol{l} = (l_1, ..., l_k)', \ \boldsymbol{u} = (u_1, ..., u_k)'). \tag{2}$$

Note that we assume that the problem uses expected values instead of probabilities such as $PR[w_{h'}(\boldsymbol{x}) > c_{h'}] \leq 0.10$; classic Kriging also assumes expected values instead of probabilities (Kriging with probabilities

is referenced in Kleijnen (2015, p. 222)). In practice, the analysts and their clients cannot observe the true $E[w_h(\boldsymbol{x})]$; they can observe the actual averages during a finite period (or short term); in the long term, "we are all dead" (as the economist Keynes said) and the system and its environment will change so the analysts should find a new solution. However, for the toy problem detailed in Section 5, we do know the true means $E[w_h(\boldsymbol{x})]$; i.e., we do know whether the estimated optimal solution is feasible. In practice, post-processing may obtain so many replications of the (relatively expensive) simulation that the analysts are (say) 95% certain that the expected values are estimated within 1%.

We denote the *gradient* of $w_h(\boldsymbol{x})$ ($h = 0, 1, ..., t-1$) at $\boldsymbol{x}_*$ by $\nabla_h(\boldsymbol{x}_*) = (\partial w_h/\partial x_1, ..., \partial w_h/\partial x_k)'|_{\boldsymbol{x}_*}$. A constraint is *binding* (or "active") at a specific $\boldsymbol{x}_*$ if this $\boldsymbol{x}_*$ lies at the boundary between the feasible area and the infeasible area; i.e.,. if this $\boldsymbol{x}_*$ changes $\leq$ into $=$ in (1), so the slack becomes exactly zero. Let $A_\lambda(\boldsymbol{x}_*)$ denote the *index set* with the indices h'' of the *binding output* constraints at $\boldsymbol{x}_*$ (obviously, this set is empty if there are no binding output constraints at this $\boldsymbol{x}_*$).

Note that the box constraint in (2) implies the $2k$ *input constraints* $x_j \leq u_j$ and $-x_j \leq -l_j$. For these constraints we define $g = 1, ..., v = 2k$, so ∇_g is a k-dimensional vector with 1 or -1 at position j, and 0 at the $k-1$ remaining positions. In our toy example, all input constraints are box constraints; however, in our (s, S) inventory example there is the additional constraint $s \leq S$ or $s - S \leq 0$, so the corresponding gradient is $(1, -1)'$.

Analogous to $A_\lambda(\boldsymbol{x}_*)$, let $A_\mu(\boldsymbol{x}_*)$ denote the *index set* with the indices g' of the *binding input* constraints at $\boldsymbol{x}_*$. Denote the *Lagrangian* multiplier for binding output constraint h'' by $\lambda_{h''}$, and for binding input constraint g' by $\mu_{g'}$. Altogether, the *KT stationarity conditions* are

$$-\nabla_0(\boldsymbol{x}_*) = \sum_{h'' \in A_\lambda(\boldsymbol{x}_*)} \lambda_{h''}(\boldsymbol{x}_*)\nabla_{h''}(\boldsymbol{x}_*) + \sum_{g' \in A_\mu(\boldsymbol{x}_*)} \mu_{g'}(\boldsymbol{x}_*)\nabla_{g'}(\boldsymbol{x}_*)$$
$$\text{with } \lambda_{h''}(\boldsymbol{x}_*) \geq 0 \text{ and } \mu_{g'}(\boldsymbol{x}_*) \geq 0. \tag{3}$$

Obviously, (3) is an *orthogonal projection* of $-\nabla_0$ onto the linear space spanned by $\nabla_{h''}$ and $\nabla_{g'}$. Because LS is also an orthogonal projection, we interpret (3) as a LS model with the estimates $\widetilde{\lambda}_{h''}$ and $\widetilde{\mu}_{g'}$ where we use a tilde to denote LS (LS is a mathematical criterion; in the next sections, we shall use a hat to denote an estimator that uses a statistical criterion such as likelihood or bias resulting in a *maximum likelihood estimator* (MLE) or an unbiased estimator). We define the vectors $\boldsymbol{\lambda}_{h''} = (\lambda_{h''})$ and $\boldsymbol{\mu}_{g'} = (\mu_{g'})$ where the number of elements of these vectors is $\#A_\lambda(\boldsymbol{x}_*)$ and $\#A_\mu(\boldsymbol{x}_*)$. Next we define $\boldsymbol{\nu} = (\boldsymbol{\lambda}'_{h''}, \boldsymbol{\mu}'_{g'})'$, and $\boldsymbol{\Delta} = (\nabla_{h''}, \nabla_{g'})$. Using (3), we define the LS model with the explained (dependent) variable $-\nabla_0$ and the

explanatory (independent) variables $\mathbf{\Delta}$. This model gives the LS estimator $\widetilde{\boldsymbol{\nu}} = (\widetilde{\boldsymbol{\lambda}}'_{h''}, \widetilde{\boldsymbol{\mu}}'_{g'})'$:

$$\widetilde{\boldsymbol{\nu}} = (\mathbf{\Delta}'\mathbf{\Delta})^{-1}\mathbf{\Delta}'(-\nabla_0). \tag{4}$$

This $\widetilde{\boldsymbol{\nu}}$ gives the LS model of the KT conditions defined in (3):

$$-\widetilde{\nabla}_0(\boldsymbol{x}_*) = \sum_{h'' \in A_\lambda(\boldsymbol{x}_*)} \widetilde{\lambda}_{h''}(\boldsymbol{x}_*)\nabla_{h''}(\boldsymbol{x}_*) + \sum_{g' \in A_\mu(\boldsymbol{x}_*)} \widetilde{\mu}_{g'}(\boldsymbol{x}_*)\nabla_{g'}(\boldsymbol{x}_*)$$
$$\text{with } \widetilde{\lambda}_{h''}(\boldsymbol{x}_*) \geq 0 \text{ and } \widetilde{\mu}_{g'}(\boldsymbol{x}_*) \geq 0. \tag{5}$$

To quantify the *validity* of this model, we use $\cos[-\nabla_0(\boldsymbol{x}_*),\ -\widetilde{\nabla}_0(\boldsymbol{x}_*)]$, which we abbreviate to $\widetilde{\cos}(\boldsymbol{x}_*)$. Ideally, $-\nabla_0(\boldsymbol{x}_*)$ and $-\widetilde{\nabla}_0(\boldsymbol{x}_*)$ *coincide* so $\widetilde{\cos}(\boldsymbol{x}_*) = 1$. However, in our search for the optimum, we require only that these two vectors point into the "same" direction; i.e., $\boldsymbol{x}_*$ lies "near" a stationary point. So, we require that $0 \leq \widetilde{\cos}(\boldsymbol{x}_*) \leq 1$. Obviously, $\cos[-\nabla_0(\boldsymbol{x}_*),\ -\widetilde{\nabla}_0(\boldsymbol{x}_*)] = \cos[\nabla_0(\boldsymbol{x}_*),\ \widetilde{\nabla}_0(\boldsymbol{x}_*)]$. Finally, we use the following well-known formula (see, e.g., Kolman and Hill (2008, p. 350)) where $\bullet$ is the usual symbol for the inner product of two vectors:

$$\widetilde{\cos}(\boldsymbol{x}_*) = \frac{\nabla_0(\boldsymbol{x}_*) \bullet \widetilde{\nabla}_0(\boldsymbol{x}_*)}{||\nabla_0(\boldsymbol{x}_*)|| \times ||\widetilde{\nabla}_0(\boldsymbol{x}_*)||}. \tag{6}$$

Unfortunately, in simulation the functions $w_h(\boldsymbol{x})$ are unknown; simulation does give the *input/output* (I/O) data $(\boldsymbol{x}_i, w_{i;h;r})$. We can use these data to estimate *metamodels* that approximate the *black-box* functions $w_h(\boldsymbol{x})$. Actually we use Kriging metamodels, as we explain next.

3 Kriging in random simulation

Subsection 3.1 summarizes SK. Subsection 3.2 derives Kriging estimators of the output gradients ∇_h. Subsection 3.3 discusses choosing the number of simulation replications m_i and input combinations n.

3.1 Stochastic Kriging

We start with the formulas for the SK predictor and its variance derived in Ankenman et al. (2010). We let $e_{h;r}(\mathbf{x}_i)$ —or briefly $e_{i;h;r}$—denote the *intrinsic noise* in replication r ($r = 1, ..., m_i$) of simulation output h ($h = 0, 1, ..., t-1$) at simulated "old" input combination $\mathbf{x}_i$ ($i = 1, ..., n$). Kriging (or GP) assumes that $e_{i;h;r}$ has a Gaussian distribution with zero mean. Averaging the m_i replications gives the average simulation output $\overline{w}_h(\mathbf{x}_i)$—or briefly $\overline{w}_{i;h}$—and the average intrinsic noise $\overline{e}_h(\mathbf{x}_i)$—or $\overline{e}_{i;h}$. Let $\mathbf{\Sigma}_{\overline{e};h}$

denote the *covariance matrix* of $\overline{e}_{i;h}$. IRN imply that this matrix is a diagonal matrix. We ignore the covariances between the t simulation outputs when fitting the t Kriging metamodels; i.e., we use *univariate Kriging* instead of multivariate Kriging or co-Kriging (Ankenman et al. (2010) considers only univariate SK). Actually, univariate Kriging drastically simplifies the analysis and may be superior, because multivariate Kriging requires the specification of an appropriate covariance matrix and the estimation of the individual covariances in that matrix; see Kleijnen and Mehdad (2014). Let $M_h(\mathbf{x})$ denote the *extrinsic noise* in the Kriging model for output h. Kriging assumes that e is independent of M. These assumptions give the following SK *predictor* for the *new* point $\boldsymbol{x}_*$ (input combination not yet simulated), which uses the n *old* simulation I/O data $(\mathbf{X}, \overline{\mathbf{w}}_h)$ where $\mathbf{X} = \boldsymbol{X}_{n\times k}$ and $\overline{\mathbf{w}}_h = \overline{\mathbf{w}}_{h;n}$:

$$\widehat{y}_h(\boldsymbol{x}_*) = \widehat{\mu}_h + \widehat{\boldsymbol{\sigma}}_{M;h}(\boldsymbol{x}_*)'(\widehat{\boldsymbol{\Sigma}}_{M;h} + \widehat{\boldsymbol{\Sigma}}_{\overline{e};h})^{-1}(\overline{\mathbf{w}}_h - \widehat{\mu}_h \mathbf{1}_n), \tag{7}$$

which *plugs-in* the MLE $\widehat{\boldsymbol{\psi}}_h = \widehat{\boldsymbol{\psi}}(\mathbf{X}, \overline{\mathbf{w}}_h)$ of $\boldsymbol{\psi}_h$ where $\boldsymbol{\psi}_h$ denotes the vector with the SK parameters including the mean $\mu_h = E(y_h)$ and the variance $\tau_h^2 = Var(y_h)$ so the $n \times n$ correlation matrix $\mathbf{R}_h = (\rho_{i;i';h})$ equals $\tau_h^{-2}\boldsymbol{\Sigma}_{M;h}$ with the $n \times n$ covariance matrix $\boldsymbol{\Sigma}_{M;h} = (\text{Cov}(y_{h;i}, y_{h;i'}))$ where $y_{h;i} = y_h(\boldsymbol{x}_i)$ and $i' = 1, ..., n$, and the n-dimensional correlation vector $\boldsymbol{\rho}_h(\boldsymbol{x}_*) = \tau_h^{-2}\boldsymbol{\sigma}_{M;h}(\boldsymbol{x}_*)$ with the n-dimensional covariance vector $\boldsymbol{\sigma}_{M;h}(\boldsymbol{x}_*) = (\sigma_{h;*}) = (\text{Cov}(y_{h;*}, y_{h;i})$ with $y_{h;*} = y_h(\boldsymbol{x}_*)$; $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$ is the classic unbiased estimator (defined in (9)); $\mathbf{1}_n$ denotes the n-dimensional vector with all elements equal to 1; $M(\boldsymbol{x})$ denotes a zero-mean stationary GP (see the subscript M of $\boldsymbol{\Sigma}_{M;h}$).

The estimated MSPE of $\widehat{y}_h(\boldsymbol{x}_*)$ is

$$\begin{aligned} s^2[\widehat{y}_h(\boldsymbol{x}_*)] &= \widehat{\tau}_h^2 - \widehat{\tau}_h^4 \boldsymbol{\rho}_h'(\boldsymbol{x}_*)[\widehat{\tau}_h^2 \mathbf{R}_h + \widehat{\boldsymbol{\Sigma}}_{\overline{e};h}]^{-1}\boldsymbol{\rho}_h(\boldsymbol{x}_*) \\ &+ \widehat{\delta}_h^2[\mathbf{1}_n'(\widehat{\tau}_h^2 \mathbf{R}_h + \widehat{\boldsymbol{\Sigma}}_{\overline{e};h})^{-1}\mathbf{1}_n]^{-1} \\ &\text{with } \widehat{\delta}_h = 1 - \mathbf{1}_n'(\widehat{\tau}_h^2 \mathbf{R}_h + \widehat{\boldsymbol{\Sigma}}_{\overline{e};h})^{-1}\boldsymbol{\rho}_h(\boldsymbol{x}_*)\widehat{\tau}_h^2. \end{aligned} \tag{8}$$

Our equation (8) equals (27) in Ankenman et al. (2010)—apart from $\widehat{\boldsymbol{\delta}}$ in the latter equation, which is actually a scalar $\widehat{\delta}$. Our equation also resembles (9) in Jalali et al. (2017).

Assuming a *constant* intrinsic variance σ_h^2 is unrealistic, so we allow $\boldsymbol{x}$ to affect both the mean and the variance of the distribution of $w_h(\boldsymbol{x})$; i.e., the intrinsic error has *heterogeneous* variances $\sigma_h^2(\boldsymbol{x}) = var[w_h(\boldsymbol{x})]$.

Following Ankenman et al. (2010), we estimate $var[\overline{w}_{i;h}]$ through

$$s^2(\overline{w}_{i;h}) = \frac{s^2(w_{i;h})}{m_i} \text{ with } s^2(w_{i;h}) = \frac{\sum_{r=1}^{m_i}(w_{i;h;r} - \overline{w}_{i;h})^2}{m_i - 1}$$
$$\text{with } \overline{w}_{i;h} = \frac{\sum_{r=1}^{m_i} w_{i;h;r}}{m_i}. \tag{9}$$

SK with IRN uses these $s^2(\overline{w}_{i;h})$ ($i = 1, ..., n$) on the *diagonal* of $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$ in (7) and (8). These $s^2(\overline{w}_{i;h})$ require only $m_i > 1$ ($s^2(w_{i;h})$ uses the denominator $m_i - 1$, so it is unbiased, unlike the MLE with denominator m_i). In Appendix 1 we discuss CRN and its analysis through either SK or OK.

3.2 Kriging estimators of output gradients

Because the KT conditions use the (unknown) gradients $\nabla_h(\boldsymbol{x}_*)$, we may use $\widehat{y}_h(\boldsymbol{x}_*)$ to estimate these $\nabla_h(\boldsymbol{x}_*)$. Inspired by the derivation of $\nabla[\widehat{y}_{\text{OK};h}(\boldsymbol{x}_*)]$ in Kleijnen et al. (2022) (also see Erickson et al. (2021)), we notice that in (7) all factors except $\widehat{\boldsymbol{\sigma}}_{M;h}(\boldsymbol{x}_*)$ depend only on the old I/O simulation data $(\boldsymbol{X}, \overline{\mathbf{w}}_{h;n})$, which also determine $\widehat{\boldsymbol{\psi}}_h = \widehat{\boldsymbol{\psi}}(\mathbf{X}, \overline{\mathbf{w}}_{h;n})$. So, we may write

$$\nabla[\widehat{y}_h(\boldsymbol{x}_*)] = \nabla[\widehat{\boldsymbol{\sigma}}_{M;h}(\boldsymbol{x}_*)'\mathbf{c}_h]$$
$$\text{with } \mathbf{c}_h = (\widehat{\boldsymbol{\Sigma}}_{M;h} + \widehat{\boldsymbol{\Sigma}}_{\overline{e};h})^{-1}(\overline{\mathbf{w}}_h - \widehat{\mu}_h \mathbf{1}_n). \tag{10}$$

Actually, we use the most popular type of correlation function in Kriging metamodels of simulation models; namely, the *anisotropic Gaussian correlation function* with parameter vector $\boldsymbol{\theta}_h = (\theta_{h;1}, ..., \theta_{h;k})'$ where $\theta_{h;j} \geq 0$:

$$\rho(\boldsymbol{\theta}_h, \boldsymbol{x}, \boldsymbol{x}') = \prod_{j=1}^{k} \exp\ [-\theta_{h;j}(x_j - x'_j)^2] = \exp\ [-\sum_{j=1}^{k} \theta_{h;j}(x_j - x'_j)^2]. \tag{11}$$

This correlation function implies $\boldsymbol{\psi}_h = (\mu_h, \tau_h^2, \theta_{h;1}, ..., \theta_{h;k})'$. This function and (10) give

$$\frac{\partial[\widehat{y}_h(\boldsymbol{x}_*)]}{\partial x_{*;j}} = -2\tau_h^2 \theta_{h;j} \{\Sigma_{i=1}^{n} c_{h;i}(x_{*;j} - x_{i:j}) \exp[\Sigma_{j'=1}^{k} - \theta_{h;j'}(x_{*;j'} - x_{i:j'})^2]\}, \tag{12}$$

where $c_{h;i}$ is element i ($i = 1, ..., n$) of $\mathbf{c}_h$ in (10) so (12) defines element j ($j = 1, ..., k$) of $\nabla[\widehat{y}_h(\boldsymbol{x}_*)]$.

3.3 Number of replications and input combinations

The number of replications m_i does not need to be a constant m, as we assume that the intrinsic noise varies with the simulation input $\boldsymbol{x}_i$ ($i = 1, ..., n$). Intuitively, a good *sample allocation rule* stipulates that m_i increases as $\sigma_h^2(\boldsymbol{x}_i)$ increases. Because $\widehat{y}_h(\boldsymbol{x}_*)$ uses the sample averages $\overline{w}_{h;i}$, our algorithm obtains so many

replications that these averages have the same estimated intrinsic variance. Let $m_{i;s}$ denote the available number of replications for input combination i in stage (or iteration) s. First we discuss $m_{i;s}$ for $s = 1, 2, ...$; next we discuss the initial stage $s = 0$ (we assume that readers are not confused by s denoting the estimated standard deviation in other contexts in our paper).

Let $\widehat{m}_{b;s}$ denote the estimated *desired* number of replications in stage s for input combination b—with $b = 1, ..., B$ and $1 \leq B \leq n$—"near" the estimated boundary (17). (Adding replications only to combinations near the boundary implies that our algorithm does not obtain expensive simulation replications for points "far away" from this boundary; consequently, the simulation observations not close to this boundary get small weights when predicting outputs near the boundary, as the extrinsic noise and the intrinsic noise are high).

If $B = 1$, then one replication is added to combination b; next, our algorithm uses all available observations $m_{b;s}$ to recompute $\overline{w}_{b;h;s}$ and $s^2(\overline{w}_{b;h;s})$, etc. If $B > 1$, then we let $m_{\min;s}$ denote the smallest number of actually simulated replications among the B combinations near the boundary in stage s (obviously, all t output types use the same input). Let $b_{\min;s}$ denote the input combination corresponding with $m_{\min;s}$. We then compute the maximum of $\widehat{m}_{b;s}$ over the t outputs, as follows:

$$\hat{m}_{b;s} = \max_h \lceil \frac{s^2(\bar{w}_{b;h;s})}{s^2(\bar{w}_{b_{\min;s};h;s})} \; m_{\min;s} \rceil \tag{13}$$

where $\lceil . \rceil$ is the usual symbol for the ceiling function (the floor function $\lfloor . \rfloor$ would imply that no replications are added if the B values of $s^2(\bar{w}_{b;h;s})$ differ only slightly, so our algorithm would get stuck). If (13) gives $\widehat{m}_{b;s} > m_{b;s}$, then we add a single replication to combination b. This new observation on $w_{b;h;s}$ changes $s^2(w_{b;h;s})$; therefore, we recompute $\hat{m}_{b;s}$ through (13). If none of the B combinations requires adding an observation at this stage, then our algorithm uses all available observations $m_{b;s}$.

It is well known that the variance of a variance estimator (such as $s^2(w_{b;h;s})$) may be relatively high. Therefore we start with $m_{\min;s} = 10$ in the toy example, but in the inventory example we start with $m_{\min;s} = 2$ because $E[s^2(w_{b;h;s})]$ is small

Next we discuss the *preprocessing* stage, testing whether the initial design gives acceptable Kriging models for the t outputs. We start with a small initial number of replications $m_{\min}$ and input combinations $n_{\min}$; e.g., in the toy problem, we choose $m_i = m_{\min} = 10$ and $n = n_{\min} = 6$. (Ankenman et al. (2010) gives the rule-of-thumb $m_i \geq 10$, and Wang and Ierapetritou (2018) uses $m_i = m = 50$ as default; Pearce et al. (2022) starts with 5 replications). Kleijnen and Van Beers (2022) derives a *leave-one-out cross-validation* (LOO-CV) test for deterministic simulation; now we derive a similar test for random simulation. So, we delete input combination i (with $i = 1, ..., n$) and its m_i replications for the t outputs $w_{i;h;r}$ (with $h = 0,$

..., $t-1$ and $r = 1$, ..., m_i), and compute the Kriging estimators $\widehat{y}_{-i;h}$ and $s^2(\widehat{y}_{-i;h})$ from the remaining I/O data $(\mathbf{X}_{-i}, \mathbf{W}_{-i})$; i.e., we use $(\mathbf{X}_{-i}, \mathbf{W}_{-i})$ instead of $(\mathbf{X}, \mathbf{W})$ in (7) and (8). Next, we restore the complete I/O data set $(\mathbf{X}, \mathbf{W})$, and delete another I/O combination i'—until we have left out each combination once. We construct two-sided CIs with "per comparison" error rates. Because these CIs should hold for all t output types and all n points, we apply the "Bonferroni" inequality; i.e., we divide by $n \times t$ to obtain an "experimentwise" error rate α_{E} (e.g., $\alpha_{\mathrm{E}} = 0.20$), and we use $z_{1-[\alpha_{\mathrm{E}}/(2nt)]}$, which is the $1-\alpha_{\mathrm{E}}/(2nt)$ quantile of the standard normal variable z. Finally, if

$$max_{i;h} \left[\frac{|\overline{w}_{i;h} - \widehat{y}_{i;h}|}{\sqrt{s^2(\overline{w}_{i;h}) + s^2(\widehat{y}_{i;h})}} \right] > z_{1-[\alpha_{\mathrm{E}}/(2nt)]}$$
$$\text{with } i = 1, ..., n \text{ and } h = 0, ..., t-1, \tag{14}$$

then we reject the t Kriging metamodels, and apply the allocation rule including (13). (Our preprocessing is inspired by Pedrielli et al. (2020), but differs in many details; e.g., that publication does not use (14).)

If $\boldsymbol{x}$ is subject to box constraints only (see, e.g., the toy problem), then—in this initial stage—we use the most popular design type in Kriging—namely, *Latin hypercube sampling* (LHS). If $\boldsymbol{x}$ is subject to an additional constraint such as $s \leq S$ in the (s, S) inventory simulation, then we still can use LHS—provided we *reparameterize* the problem; i.e., we use LHS with $x_1 = s$ and $x_2 = Q = S$ - s (a similar reparameterization is used in Law (2015, p. 636) for an (s, S) inventory simulation without a service-level constraint, analyzed through a local second-order polynomial instead of a global Kriging metamodel). In LHS we use *midpoints*, which is an option in MATLAB's function *lhsdesign*. This option fixes the distances among the n projected values of a specific input j (with $j = 1$, ..., k). We expect that this option gives better $\widehat{\boldsymbol{\psi}}_h$ in the anisotropic correlation function (11), because these projected values are not clustered; actually, these values lie apart from each other at distances that are multiples of $1/n$.

We start with $n = (k+1)(k+2)/2$ if $k \leq 6$ and $n = 5k$ if $k > 6$, following Tao et al. (2020)'s n for sequential optimization in deterministic simulation.

Note that Loeppky et al. (2009) proposes $n = 10k$, as a rule-of-thumb for sensitivity analysis via Kriging. This rule is also used in Wang and Ierapetritou (2018) for sequential optimization in random simulation. For $n = 5k$,Tao et al. (2020) refers to Liu et al. (2017). Pandita et al. (2019) states that $5k$ to $10k$ is used, and also refers to Sóbester et al. (2005) (the latter recommends n equal to 35% of the total computer budget, which is assumed to be fixed). For sequential optimization in deterministic simulation with $k = 2$, Pourmohamad and Lee (2021) uses $n = 20$ in the "modified Townsend" example, but uses $n = 10$ in their toy example that we shall change into a random simulation.

4 Efficient global optimization

Optimization should avoid convergence to a local optimum. Therefore, Jones et al. (1998)—the seminal publication on EGO for *unconstrained* optimization in *deterministic* simulation—derives an estimator of EI at $\boldsymbol{x}$, and uses this $\widehat{\text{EI}}(\boldsymbol{x})$ to select the next point to be simulated. For *unconstrained* optimization in *random* simulation with heterogeneous noise, Jalali et al. (2017) evaluates six variations of EI; actually, we define our estimator $\widehat{\text{MEI}}$ in (16) below. For *constrained* optimization in *deterministic* simulation, Kleijnen et al. (2022) uses an estimator of EI multiplied by an estimator of the cos defined in (6). Now we focus on *constrained* optimization in *random* simulation.

The simulation estimators $\overline{w}_{i:h}$ may be rather inaccurate, given $\sigma(\boldsymbol{x}_i)/m_i$. Therefore our algorithm uses the SK predictor $\widehat{y}_h$ (defined in (7)), which has a lower MSPE than $\overline{w}_{i:h}$ has—assuming there are "enough" old points "near" $\boldsymbol{x}_i$, and the bias of the Kriging predictor is "small". However, our algorithm may give estimated optimal solutions (say) $\widehat{\boldsymbol{x}}_{\min}$ that are actually infeasible; e.g., if $\widehat{y}_h$ has a symmetric distribution, then $PR[\widehat{y}_{h'}(\widehat{\boldsymbol{x}}_{\min}) > c_{h'}]$ may be as high as 50%. Therefore, our algorithm accepts $\widehat{\boldsymbol{x}}_{\min}$ as the optimal input combination only if

$$\widehat{y}_{\,h'}(\widehat{\boldsymbol{x}}_{\min}) + z_{(1-\alpha)} s[\widehat{y}_{h'}(\widehat{\boldsymbol{x}}_{\min})] \leq c_{h'} \text{ with } \alpha = \alpha_{\text{infe}} \tag{15}$$

where α_{infe} is specified by the users to control the probability of an infeasible solution selected by our algorithm ($s(\widehat{y}_{h'})$ is the square root of $s^2(\widehat{y}_{h'})$ defined in (8)). For example, if α_{infe} is 50%, 10%, 1%, or 0.1%, then the "safety factor" $z_{(1-\alpha)}$ in (15) is 0.0000, 1.2816, 2.3263, or 3.0902.

Note that $s^2(\widehat{y}_{h'})$ underestimates the true MSPE, because it assumes known Kriging parameters $\boldsymbol{\psi}_h$ and an unbiased predictor $\widehat{y}_{h'}$. Because (15) uses the Studentized slack $\{[\widehat{y}_{\,h'}(\widehat{\boldsymbol{x}}_{\min}) - c_{h'}]/s[\widehat{y}_{h'}(\widehat{\boldsymbol{x}}_{\min})\}$, we would like to use the Student t-statistic with (say) υ degrees of freedom t_υ. However, we do not know the correct value of υ, so we use z; of course, $z_{(1-\alpha)} < t_{\upsilon;(1-\alpha)}$.

So, our algorithm estimates the best feasible goal output by $\widehat{y}_{0;\,\min} = \min_{1 \leq i \leq n} \widehat{y}_0(\boldsymbol{x}_i)$ if (15) holds (our initial design and PS satisfy all input constraints); else $\widehat{y}_{0;\,\min} = \infty$. This $\widehat{y}_{0;\,\min}$ gives

$$\widehat{\text{MEI}}(\boldsymbol{x}) = (\widehat{y}_{0;\,\min} - \widehat{y}_0(\boldsymbol{x}))\, \Phi\left(\frac{\widehat{y}_{0;\,\min} - \widehat{y}_0(\boldsymbol{x})}{s[\widehat{y}_0(\boldsymbol{x})]}\right) + s[\widehat{y}_0(\boldsymbol{x})]\phi\left(\frac{\widehat{y}_{0;\,\min} - \widehat{y}_0(\boldsymbol{x})}{s[\widehat{y}_0(\boldsymbol{x})]}\right). \tag{16}$$

Obviously, $\widehat{\text{MEI}}(\boldsymbol{x})$ is positive, but it may become so small that we stop our search; actually, our algorithm stops as soon as $\widehat{\text{MEI}}(\boldsymbol{x}) < \varepsilon \times |\widehat{y}_{0;\,\min}|$ where ε is some small positive number; e.g., 0.01.

To estimate how well the KT conditions hold at $\boldsymbol{x}$, we need to estimate which output constraints are *binding* at $\boldsymbol{x}$. Therefore we use the following decision rule:

$$\text{if } \frac{|\widehat{y}_{h'}(\boldsymbol{x}) - c_{h'}|}{s[\widehat{y}_{h'}(\boldsymbol{x})]} \leq z_{1-\alpha/2}, \text{ then } h' \in A_\lambda(\boldsymbol{x}). \tag{17}$$

Our algorithm searches for the solution (say) $\widehat{\boldsymbol{x}}_{\text{o}}$ of

$$\max_{\boldsymbol{x}} \widehat{\text{MEI}}(\boldsymbol{x}) \times \cos\{\widehat{\nabla}[\widehat{y}_0(\boldsymbol{x})], \widetilde{\widehat{\nabla}}[\widehat{y}_0(\boldsymbol{x})]\}, \tag{18}$$

while satisfying the $\#A_\lambda(\boldsymbol{x})$ binding output constraints in (17) and all v input constraints in (1); in (18) $\cos\{\widehat{\nabla}[\widehat{y}_0(\boldsymbol{x})], \widetilde{\widehat{\nabla}}[\widehat{y}_0(\boldsymbol{x})]\}$ equals

$$\widetilde{\widehat{\cos}}[\boldsymbol{x}] = \frac{\widehat{\nabla}[\widehat{y}_0(\boldsymbol{x})] \bullet \widetilde{\widehat{\nabla}}[\widehat{y}_0(\boldsymbol{x})]}{||\widehat{\nabla}[\widehat{y}_0(\boldsymbol{x})]|| \times ||\widetilde{\widehat{\nabla}}[\widehat{y}_0(\boldsymbol{x})]||} \tag{19}$$

where $\widetilde{\widehat{\nabla}}[\widehat{y}_0(\boldsymbol{x})]$ is the LS estimator

$$-\widetilde{\widehat{\nabla}}[\widehat{y}_0(\boldsymbol{x})] = \sum_{h'' \in A_\lambda(\boldsymbol{x})} \widetilde{\widehat{\boldsymbol{\lambda}}}_{h''}(\boldsymbol{x}) \widehat{\nabla}[\widehat{y}_{h''}(\boldsymbol{x})] + \sum_{g' \in A_\mu(\boldsymbol{x})} \widetilde{\mu}_{g'}(\boldsymbol{x}) \nabla_{g'}(\boldsymbol{x})$$
$$\text{with } \widetilde{\widehat{\boldsymbol{\lambda}}}_{h''}(\boldsymbol{x}) \geq 0 \text{ and } \widetilde{\mu}_{g'}(\boldsymbol{x}) \geq 0. \tag{20}$$

Note that deterministic simulation uses EI so EGO does not select old points $\boldsymbol{x}_i$ (because $\widehat{\text{EI}}(\boldsymbol{x}_i) = 0$) for the next simulation, whereas random simulation may give $\widehat{\text{MEI}}(\boldsymbol{x}_i) > 0$ so our algorithm may revisit old points. More precisely, if $\boldsymbol{x} = \boldsymbol{x}_i$, then the first term in (16) is not positive and (because $\widehat{y}_0(\boldsymbol{x}_i)$ is not an exact interpolator) the second term is positive.

To find the solution $\widehat{\boldsymbol{x}}_{\text{o}}$ for the problem defined in (18) through (20), we use MATLAB's *pattern search* (PS). PS is detailed on

https://nl.mathworks.com/help/gads/patternsearch.html.

That website describes several types of optimization problems, including "Nonlinear Constraints" (we quote text on the PS website so the readers can easily retrieve the relevant text). PS has several "Options", including "InitialMeshSize" and "Tolerance on the mesh size". For both options we select values that depend on the specific example; i.e., for the toy problem with $0 \leq x_j \leq 1$ ($j = 1, 2$), we select the "InitialMeshSize" equal to 0.10 (or 10% of the input range), and for the inventory problem with $600 \leq s \leq 1{,}200$ and $600 \leq S \leq 1{,}800$ we select the value 20. During its search, PS decreases (or increases) this mesh size when the next iterate is worse (or better) than the current iterate (i.e., PS uses an "adaptive mesh"). "Tolerance on the

mesh size" stops the PS search as soon as the mesh size becomes smaller than this tolerance. For the toy problem with $0 \leq x_j \leq 1$, we select a tolerance equal to 0.001, and for the inventory problem we select a tolerance of 5. PS also has an "exitflag — Reason patternsearch stopped"; e.g., the flag value -2 means "No feasible point found". (We do not use the option "UseParallel" in PS; in our algorithm we do use parallel computing.) PS requires an "Initial point". To reduce the probability of PS getting *trapped* in a local optimum, PS may be *restarted* several times. Actually, we use $10k$ starting points that satisfy the v input constraints $f_g(\boldsymbol{x}) \leq c_g$ $(g = 1, ..., v)$, but may violate the output constraints (MATLAB advises ten feasible restart points). PS requires relatively little computer time in our algorithm, as PS uses $\widehat{y}_h$ instead of w_h (expensive simulation output). To select specific initial (so, $s = 0$) starting points, we use LHS. Obviously, PS also ends with $10k$ points, but some of these endpoints may coincide. To select *specific* starting points after the initial stage (so, $s = 1, 2, ...$), we also use the new points that were selected by our algorithm after stage $s = 0$. However, in the first stages, there are not enough new points, so PS also uses LHS. (Wang and Ierapetritou (2018) also uses $10k$ starting points selected by LHS, but uses MATLAB's fmincon to find infill points).

If PS cannot find a feasible solution (exitflag is -2), then we halve α and apply PS to the problem with this new α-value (obviously, this new value implies a wider CI around the estimated binding output constraints). We stop our algorithm as soon as the current α is lower than some small value (say) $\alpha_{\min}$; e.g., $\alpha_{\min} = 0.01$.

After PS has given $\widehat{\boldsymbol{x}}_{\text{o}}$, our algorithm must decide on m_i. Our algorithm applies the sample allocation rule (13); i.e., it may add (expensive) *replications*, at old or new points. Next, it uses all available simulated I/O combinations, to re-estimate the SK parameters.

Whereas deterministic simulation gives the true outputs $w_{i;h}$, random simulation gives only the estimates $w_{i:h;r}$ $(r = 1, ..., m_i)$. Consequently, random simulation may require many more iterations, and still give an estimated global optimum that lies far away from the true optimum. Therefore our algorithm *restarts* with different initial designs (in general, restarting is popular in optimization). We use 12 restarts (actually, at the start of our experimentation we used a non-parallel computer; we may then add a single restart at a time—until we are satisfied with the result or we run out of time). Our algorithm uses PS with $n = 10k$ restarts, not in parallel. (In practice, the users may observe the development of the estimated optimum, and decide whether they wish to continue the search for the true optimum.)

We use the following hardware: an HPE ProLiant DL380 Gen10 server with two Intel(R) Xeon(R) Silver 4110 CPUs (16 cores) with 2.10 GHz, and 59.1 GB of usable RAM out of 64 GB. We use MATLAB R2019b; also see the Parallel Computing Toolbox on

https://www.mathworks.com/help/parallel-computing/parallel-preferences.html.

We use the PRN generator called *mrg32k3a*, which can create 2^{63} independent PRN substreams of length

2^{127}; see

https://www.mathworks.com/help/matlab/ref/randstream.randstream.create.html.

This generator is also recommended in L'Ecuyer (2015). This generator enables us to control the PRN seeds such that restarts and macroreplications use non-overlapping substreams.

Note that initially we used a non-parallel implementation without restarting (or 1 "restart" instead of 12 restarts). In the toy example, this implementation gave many final values for $\widehat{\boldsymbol{x}}_{\min}$ close to a local optimum instead of the global optimum. More precisely, in this example 60 macroreplications without restarts gave frequency diagrams for $\widehat{y}_{0;\,\min}$ with two "tops" (or modes), which we interpret as tops corresponding with the global and local optima.

In Algorithm 1, we present a pseudo-code for a single restart.

Algorithm 1

1. Select the algorithm's control variables α_{E}, α, α_{infe}, $\alpha_{\min}$, ε, m, n.

   Comment: The variable α_{E} features in (14), α and α_{infe} in (15), $\alpha_{\min}$ after (20), ε below (16), and m and n in Section 3.3.

2. Sample an initial design matrix with n combinations of the k simulation inputs x_j that satisfies the input constraints $f_g(\boldsymbol{x}) \leq c_g$ with $g = 1, ..., v$.

   Comment: This Step gives the $n \times k$ matrix $\boldsymbol{X}$ with the k-dimensional vectors $\boldsymbol{x}_i$ where $i = 1, ..., n$. We select $n = 6$ if $k = 2$.

3. Use $\boldsymbol{X}$ as input for the given simulation model, to obtain m replications of the $n \times t$ matrix with (t-variate) simulation outputs $\mathbf{W}_{n\times t;r}$.

   Comment: $\mathbf{W}_{n\times t;r}$ consists of the n-dimensional vectors $\mathbf{w}_{n;h;r}$ with $h = 0, ..., t-1$ and $r = 1, .., m$. We select the initial value $m = 10$ in the toy example and $m = 2$ in the inventory example.

4. Apply LOO-CV to $(\boldsymbol{X}, \mathbf{W}_{n\times t;r})$.

   Comment: LOO-CV uses (14).

5. If LOO-CV rejects one or more SK models, then obtain a new replication for the combination with the smallest estimated variance among the t output types, and apply the allocation rule to compute the desired number of replications $\widehat{m}_i$; if necessary, then obtain additional replications, and update m_i; return to Step 4.

   Comment: To compute $\widehat{m}_i$, use (13). To decide on additional replications, use the text below (13). Steps 4 and 5 constitute the preprocessing stage.

6. Use $(\boldsymbol{X}, \mathbf{W}_{n\times t;r})$ to compute $\widehat{\boldsymbol{\psi}}_h = \widehat{\boldsymbol{\psi}}(\boldsymbol{X}, \mathbf{w}_{n;h;r})$, $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$, $\widehat{y}_h(\boldsymbol{x}_i)$, $s^2[\widehat{y}_h(\boldsymbol{x}_i)]$.
   Comment: $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$ is a diagonal matrix that uses (9).

7. Use $\widehat{y}_h(\boldsymbol{x}_i)$ to compute $\widehat{y}_{0;\,\min} = \min_i\, \widehat{y}_0(\boldsymbol{x}_i)$ if $\widehat{y}_{h'}(\boldsymbol{x}_i) + z_{(1-\alpha)}s[\widehat{y}_{h'}(\boldsymbol{x}_i)] \leq c_{h'}$ (with $h' = 1, ..., t-1$ and $\alpha = \alpha_{\text{infe}}$) and $f_g(\boldsymbol{x}) \leq c_g$ (with $g = 1, ..., v$); else $\widehat{y}_{0;\,\min} = \infty$.
   Comment: This $\widehat{y}_{0;\,\min}$ is needed to compute $\widehat{\text{MEI}}(\boldsymbol{x})$, in Step 9.

8. Use $\widehat{y}_h(\boldsymbol{x}_i)$ to compute $\widehat{\nabla}[\widehat{y}_h(\boldsymbol{x}_i)]$.
   Comment: To compute $\widehat{\nabla}[\widehat{y}_h(\boldsymbol{x}_i)]$, use (12). This $\widehat{\nabla}[\widehat{y}_h(\boldsymbol{x}_i)]$ is needed to compute $\widetilde{\cos}[\boldsymbol{x}]$, in Step 9.

9. Apply PS to estimate $\widehat{\boldsymbol{x}}_{\text{o}}$, which maximizes $\widehat{\text{MEI}}(\boldsymbol{x}) \times \widetilde{\cos}[\boldsymbol{x}]$ such that $\boldsymbol{x}$ satisfies the $t-1$ CIs for the constrained outputs h' defined in (17) and the v input constraints defined in (1).
   Comment: $\widehat{\text{MEI}}(\boldsymbol{x})$ is defined in (16) using $\widehat{y}_0$ and $s^2(\widehat{y}_0)$; $\widetilde{\cos}[\boldsymbol{x}]$ is defined in (19) using the estimated gradients of the estimated binding output constraints $\widehat{\nabla}[\widehat{y}_{h''}(\boldsymbol{x})]$ and the LS model with the explained variable $\widetilde{\widehat{\nabla}}[\widehat{y}_0(\boldsymbol{x})]$ and the explanatory variables $\widehat{\nabla}[\widehat{y}_{h''}(\boldsymbol{x})]$ and $\nabla_{g'}(\boldsymbol{x})$—defined in (20).

10. If PS finds a feasible point $\widehat{\boldsymbol{x}}_{\text{o}}$, then apply the allocation rule, obtain the required observations, and return to Step 6; else replace α by $\alpha/2$ in (17). If $\alpha \geq \alpha_{\min}$, then return to Step 9; else if $\alpha < \alpha_{\min}$ and $\widehat{\text{MEI}}(\boldsymbol{x}) < \varepsilon \times |\widehat{y}_{0;\,\min}|$, then go to the final step (Step 11).

11. Present the estimated optimal goal output $\widehat{y}_{0;\,\min}$.

After the multiple (namely, 12) *restarts*, our algorithm presents the minimum of the multiple estimated optimal goal outputs. Let $\widehat{y}_{0;\,\min;q}$ denote $\widehat{y}_{0;\,\min}$ of Step 11 in Algorithm 1 (single restart) of restart q (our PC with 12 cores implies $q = 1, ..., 12$). Then all restarts together give the *overall* estimated optimal goal output; namely, $\widehat{y}_{0;\,\min;q\text{o}}$ with $q\text{o} = \arg min_q\ \ \widehat{y}_{0;\,\min;q}$. Moreover, the algorithm determines information that corresponds with $\widehat{y}_{0;\,\min;q\text{o}}$; namely, $\widehat{\boldsymbol{x}}_{\text{o};\,q\text{o}}$, $\widehat{y}_{h';\,q\text{o}}(\widehat{\boldsymbol{x}}_{\text{o};\,q\text{o}})$ ($h' = 1, ..., t-1$), and the (say) 90% two-sided CI for $\widehat{y}_{h;\,q\text{o}}(\widehat{\boldsymbol{x}}_{\text{o};\,q\text{o}})$ ($h = 0, ..., t-1$). The confidence levels are supposed to apply per comparison; however, they are not exact, as the algorithm has many steps that use CIs. Because of the KT conditions, the estimated slacks $c_{h'}$ - $\widehat{y}_{h'\,q\text{o}}(\widehat{\boldsymbol{x}}_{\text{o};\,q\text{o}})$ should be virtually zero for at least one output constraint.

Our algorithm uses 12 restarts (cores), so we select the following measure of *efficiency*. By definition, the *elapsed time* is the time between the starting time and the stopping time of our algorithm. That stopping time is determined by the *slowest* restart among the 12 restarts, which requires the maximum number of (expensive) simulation observations (the adapted EGO computations require negligible time). Restart q (with $q = 1, ..., 12$) stops after T_q stages (so, $s = 0, 1, ..., T_q$). Stage s has its own number of simulated points n_s, and each of these points has its own number of replications $m_{i;s}$. Let N_q denote the *total* number

of simulation observations of restart q, and let N denote N_q of the slowest restart:

$$N = \max_q N_q \text{ with } N_q = \sum_{s=0}^{T_q} \sum_{i=1}^{n_s} m_{i;s;q} \tag{21}$$

We use this N as a *pseudo-measure* for the elapsed time between the simultaneous start of the 12 restarts and the end of the slowest restart.

Note that we may add replications to either the B points near the boundary or all n points. We conjecture that both design options are equally effective, but that the former design option is more efficient. To test this conjecture, we compare these two options in the toy example, using one macroreplication with 12 restarts. Adding replications to all n points gives $\widehat{\boldsymbol{x}}_{\min} = (0.1938, 0.4187)'$ with $\widehat{y}_{0;\,\min} = 0.6159$, and N = 101,671. Adding replications to B points gives $\widehat{\boldsymbol{x}}_{\min} = (0.1984, 0.4115)'$ with $\widehat{y}_{0;\,\min} = 0.6122$ (nearly 1% better) and N = 47,707 (less than 50%).

Inspired by Tao et al. (2020, Table 1, line 32), we add an *extra Step* that finds the solution $\widehat{\boldsymbol{x}}_{\min}$ for

$$\begin{aligned} &\min_{\boldsymbol{x}} \widehat{y}_0(\boldsymbol{x}) \text{ subject to} \\ &\widehat{y}_{h'}(\boldsymbol{x}) + z_{(1-\alpha)} s[\widehat{y}_{h'}(\boldsymbol{x})] \leq c_{h'} \text{ with } \alpha = \alpha_{\text{infe}} \\ &f_g(\boldsymbol{x}) \leq c_g \;\; g = 1, ..., v \end{aligned} \tag{22}$$

where $\widehat{y}_h$ (h = 0, ..., $t-1$) uses the simulation I/O data of all 12 restarts. To find this $\widehat{\boldsymbol{x}}_{\min}$, our algorithm uses PS with 12 restarts in parallel. Unlike Tao et al., we check whether simulation with this input $\widehat{\boldsymbol{x}}_{\min}$ gives a feasible solution and a smaller $\widehat{y}_0(\widehat{\boldsymbol{x}}_{\min})$; else $\widehat{\boldsymbol{x}}_{\min}$ equals $\widehat{\boldsymbol{x}}_{\text{o};\,q\text{o}}$ (estimate resulting from multiple restarts of our algorithm).

Note that these 12 restarts may give so many simulated combinations that a numerical problem arises (DACE gives the error message "Multiple design sites are not allowed"). To solve this problem, we combine I/O simulation data that lie "close together"; e.g., if the toy example (with $0 \leq x_{i;j} \leq 1$) gives $|x_{i;j} - x'_{i;j}| < 0.07$, then we replace these inputs and outputs by their averages $\overline{x}_{i;j}$ and $\overline{w}_i$.

5 Mathematical toy example

We display the deterministic toy example of Gramacy et al. (2016) (also used in several other publications mentioned in Section 1) in Fig. 1, which includes the following special points: A is the global minimum; B is a local minimum; C is a local minimum with a binding input constraint; D and E are local maxima; A

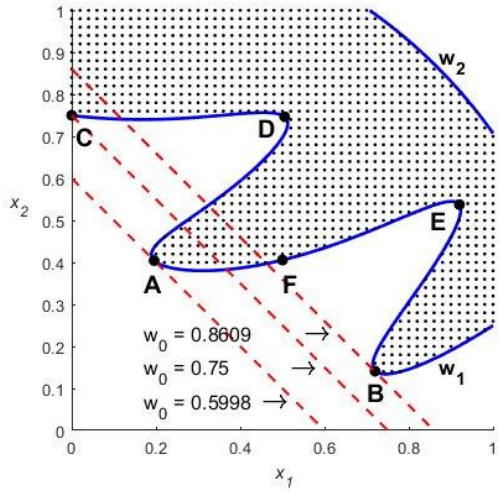

Figure 1: Mathematical toy example with inputs x_1 and x_2, deterministic goal output w_0 and constraints w_1 and w_2, and special points A through F

through E satisfy the KT conditions, whereas F does not; A through F are investigated in Kleijnen et al. (2022).

Now we modify this deterministic example such that it becomes relevant for random simulation; i.e., we introduce *intrinsic noise* $e_h(\boldsymbol{x})$ with $h = 0, 1, 2$, and replace the deterministic outputs $w_h(\boldsymbol{x})$ by their *expected values* $E[w_h(\boldsymbol{x})]$:

$$\begin{aligned}
\min_{\boldsymbol{x}} E[w_0(\boldsymbol{x})] &= E[x_1 + x_2 + e_0(\boldsymbol{x})] \\
E[w_1(\boldsymbol{x})] &= E\{\frac{3}{2} - x_1 - 2x_2 - \frac{1}{2}\sin[2\pi(x_1^2 - 2x_2)] + e_1(\boldsymbol{x})\} \leq 0 \\
E[w_2(\boldsymbol{x})] &= E[-\frac{3}{2} + x_1^2 + x_2^2 + e_2(\boldsymbol{x})] \leq 0 \\
0 \ &\leq \ x_j \leq 1 \ j = 1, 2.
\end{aligned} \tag{23}$$

We assume that e_h has *heterogeneous variances* that are determined by $\boldsymbol{x}$; so, we let $\sigma^2_{e;h}(\boldsymbol{x})$ denote these variances. To select specific values for these $\sigma^2_{e;h}(\boldsymbol{x})$, we follow Jalali et al. (2017, p. 293) where the "best case" means that the standard deviation $\sigma_{e;h}(\boldsymbol{x})$ decreases as $E[w_h(\boldsymbol{x})]$ decreases, and the "worst" case means that $\sigma_{e;h}(\boldsymbol{x})$ increases as $E[w_h(\boldsymbol{x})]$ decreases; Jalali et al. (2017) defines these two cases, assuming no constraints when minimizing $E[w_0(\boldsymbol{x})]$. We, however, consider multiple outputs. For simplicity's sake, we apply Jalali et al.'s approach to each of these outputs. We limit our experiments to the best case (so, as

$w_h(\boldsymbol{x})$ decreases, $\sigma_{e;h}(\boldsymbol{x})$ decreases) with relatively small noise:

$$\begin{aligned}\sigma_{e;0}(\boldsymbol{x}) &= 0.30 + 0.45E[w_0(\boldsymbol{x})] \\ \sigma_{e;1}(\boldsymbol{x}) &= 1.1507 + 0.45E[w_1(\boldsymbol{x})] \\ \sigma_{e;2}(\boldsymbol{x}) &= 0.975 + 0.45E[w_2(\boldsymbol{x})].\end{aligned} \tag{24}$$

These experiments still require many simulation observations; e.g., macroreplication # 1 requires $N = 47{,}707$. Appendix 2 details the derivation of formulas for small noise versus big noise in the best case versus the worst case.

For simplicity's sake, we assume *zero cross-correlations* between different types of output in the toy example (the inventory example in Section 6 has two negatively correlated output types). To sample the univariate $e_h(\boldsymbol{x})$, we use MATLAB's *normrnd* where $E[e_h(\boldsymbol{x})] = 0$ and $\sigma_{e;h}(\boldsymbol{x})$ follows from (24). Given these $e_h(\boldsymbol{x})$, we compute the corresponding w_h ($h = 0, 1, 2$) using (23). Appendix 4 shows how our algorithm proceeds through various stages.

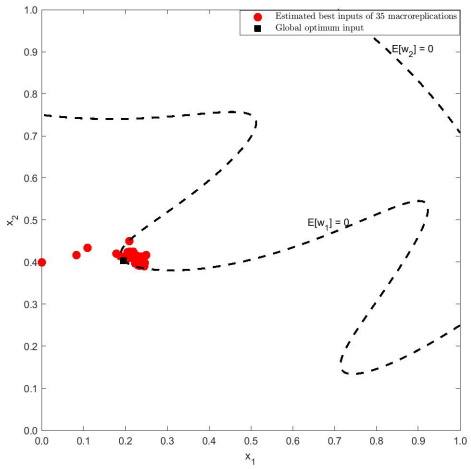

Figure 2: Final estimated optimal input combinations $\widehat{\boldsymbol{x}}_{\min}$ in toy problem with 35 macroreplications, each with 12 restarts, $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, and 20 PS starting points

Fig. 2 presents $\widehat{\boldsymbol{x}}_{\min}$, which denotes the estimated optimal input combinations after 12 restarts, in 35 macroreplications; each macroreplication uses $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, and 20 PS starting points in each iteration. We add that 5 of these 35 estimates result from the extra step at the end of our algorithm; see (22). This Figure clearly shows that 3 of these 35 macroreplications are actually infeasible—which agrees with $\alpha_{\text{infe}} = 10\%$. The other 32 macroreplications give estimates close to the true optimum.

The left-hand side of Fig. 3 presents the boxplot of the final estimated optimal goal outputs $\hat{y}_0(\hat{\mathbf{x}}_{\min})$ in

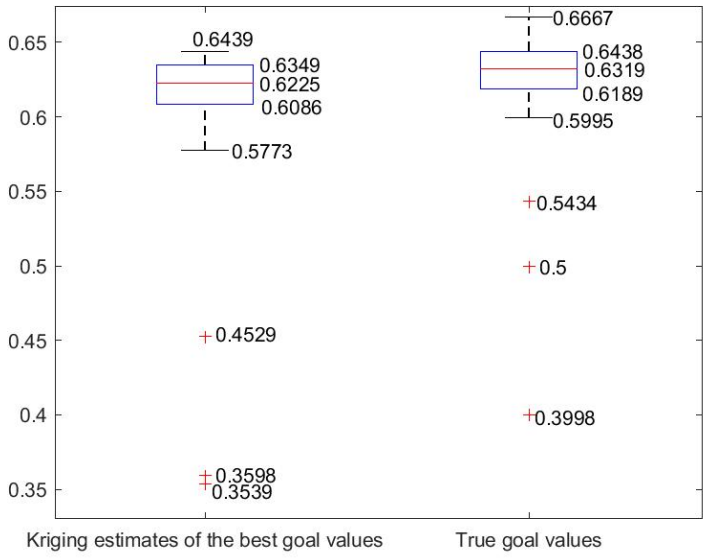

Figure 3: Boxplot of $\hat{y}_{\text{SK};0}(\widehat{\boldsymbol{x}}_{\min})$ (left-hand side) and $E[w_0(\hat{\mathbf{x}}_{\min})]$ (right=hand side) in toy problem; 35 macroreplications, 12 restarts, $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, and 20 PS starting points

35 macroreplications (12 restarts, $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, 20 PS starting points). The three outliers below the lowest quartile correspond with the three infeasible input combinations in Fig. 2. The sample median is 0.6225 (the true optimum is 0.5998, so the estimate is nearly 4% higher). The right-hand side presents the boxplot of $E[w_0(\hat{\mathbf{x}}_{\min})]$ (we know this mean, because this example is only a mathematical toy problem).

Fig. 4 presents the boxplot for N in 35 macroreplications. Its median is $14,382$; i.e., random simulation with large response variances may require much computer time.

Note that SK combined with CRN requires the estimation of the elements of the $n \times n$ covariance matrix $\boldsymbol{\Sigma}_{\overline{e};h}$—which requires $m > n$ (see Dykstra (1970)). In the toy example, we select $n = 6$ and $m = 7$ in the preprocessing stage. Based on five macroreplications—where α_{infe} is either 0.10 or 0.01—we conclude that SK with CRN performs worse than OK with CRN. We conjecture that the cause is the estimation of the extra Kriging parameter $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$. Therefore, we focus on SK without CRN.

5.1 Our algorithm versus OptQuest in toy example

In this subsection, we numerically compare the performance of OptQuest and our algorithm. OptQuest is a black-box optimizer, as it is proprietary. Both OptQuest and our algorithm treat the simulation model as a black box; i.e., in our experiment, OptQuest receives the input $(\boldsymbol{x}_i, w_{i;h;r})$ (which is the simulation I/O with $i = 1, ..., n$; $h = 0, 1, 2$; $r = 1, ..., m_i$).

For OptQuest we select m_i through the sequential procedure in Law (2015, p. 505) for estimating the

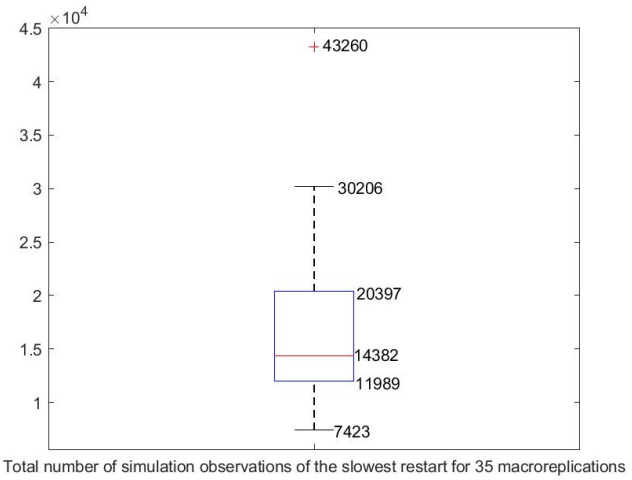

Figure 4: Elapsed times N in toy problem; 35 macroreplications, 12 restarts, $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, and 20 PS starting points

mean $E(w)$ with a relative error of γ (with $0 < \gamma < 1$) and confidence level $1-\alpha$. This procedure starts with an initial number of replications $m = m_0 \geq 10$ (we use our own symbols). Next, it computes the half-length of the two-sided $1-\alpha$ CI:

$$l = t_{m-1;1-\alpha/2} \frac{s(w)}{\sqrt{m}}. \tag{25}$$

If

$$\frac{l}{|\bar{w}|} < \frac{\gamma}{1+\gamma}, \tag{26}$$

then stop adding replications; else, add one replication, and update (25) and (26), etc.—until (26) holds. (Alternative procedures are discussed in Kleijnen (2015, pp. 107–108).)

However, we point out that if (the denominator) $|\bar{w}|$ is "close" to zero, then this procedure gives a "high" $\widehat{m}$ (desired number of replications). To solve this problem, we return to the original optimization problem defined in (1), which includes the $(t-1)$ output constraints $E[w_{h'}(\boldsymbol{x})] \leq c_{h'}$. Our toy example defined in (23), however, has the constraints $E[w_1(\boldsymbol{x})] = E\{3/2 - x_1 - ...\} \leq 0$ and $E[w_2(\boldsymbol{x})] = E[-3/2 + x_1^2 + ...] \leq 0$. To derive $\widehat{m}$, we replace $w_{h'}(\boldsymbol{x})$ by $w_{h'}(\boldsymbol{x}) + c_{h'}$ (which has the same variance as $w_{h'}(\boldsymbol{x})$ has); i.e., in the toy problem we use $w_1(\boldsymbol{x}) - 3/2$ and $w_2(\boldsymbol{x}) + 3/2$ (we do not replace $w_0(\boldsymbol{x})$). We still use $w_{h'}(\boldsymbol{x})$ when we use OptQuest to search for the optimum.

Because we have t outputs, we wish that $m \geq \widehat{m}$ holds for all t output; obviously, $\boldsymbol{x}_i$ has its own $E[w_h(\boldsymbol{x}_i)]$ and $Var[w_h(\boldsymbol{x}_i)]$. Altogether, we define the estimated *desired* number of replications for $\boldsymbol{x}_i$

$$\widehat{m}(\boldsymbol{x}_i) = \max_t \ [\widehat{m}_t(\boldsymbol{x}_i)]. \tag{27}$$

We think that our procedure for determining m agrees with one of the OptQuest options that also uses γ and α (this option is part of the OptQuest black box).

Note that our own algorithm uses the allocation rule (13), and allows the search to return to old points and add replications. More precisely, our rule selects the *relative* number of replications for different input combinations, whereas Law's rule selects the absolute number of replications for a given input combination; our rule uses the estimated variances of the simulation outputs, whereas Law's rule uses the estimated standard deviations (so it dampens the effect of variance heterogeneity).

Because (27) may require many replications if γ is small (e.g., 5%), we may select a relatively high γ (e.g., 10%). The latter selection implies a relatively high $Var[w_h(\boldsymbol{x}_i)]$, so we may then select a relatively high safety factor $z_{(1-\alpha)}$ in (15). (which defines the problem solved by OptQuest and our procedure).

We start OptQuest with the same *initial stage* as we use for our algorithm; i.e., we start OptQuest with $(\boldsymbol{x}_i, w_{i;h;r})$ with $n = 6$ input combinations selected through LHS (with midpoints), and m_i ($i = 1, ..., n$) replications where m_i follows from (27). This start is quite general; i.e., n follows from Tao et al. (2020) and these m_i are based on Law (2015), which is the most popular textbook on simulation. Our algorithm uses 12 cores (of our PC) and starts each core with its own initial $n \times k$ LHS design and initial m_i, per macroreplication. However, OptQuest does not use multiple cores (we do use our 12 cores to speed-up the simulation of the toy example). Moreover, we start each macroreplication of OptQuest with the $n \times k = 6 \times 2$ LHS design that gave the best results for our algorithm in the corresponding macroreplication (so we give OptQuest a good start in each macroreplication). Of course, we select the m_i in OptQuest using Law's procedure defined in (25) etc.

After these *initial* I/O simulation data, OptQuest asks for simulation I/O data for a specific *new* input combination $\boldsymbol{x}_{n+1}$ (which is a point "near" one or more points selected by the space-filling LHS-design). So, we provide $(\boldsymbol{x}_{n+1}, w_{n+1;h;r})$ to OptQuest, using (27) to decide on m_{n+1}.

We let OptQuest solve the analogue of (22), replacing $\widehat{y}_h(\boldsymbol{x}_i)$ by $w_{h;r}(\boldsymbol{x}_i)$. The three outputs ($w_{0;r.}(\boldsymbol{x}_i)$, $w_{1;r.}(\boldsymbol{x}_i)$, $w_{2;r.}(\boldsymbol{x}_i)$) are uncorrelated, in the toy example (in the inventory example the—possibly nonnormal—simulation outputs are statistically dependent). By definition, replications are IID. Furthermore, the simulation outputs of the n combinations are independent because the simulation does not use CRN). Each of the 12 cores of our PC works in parallel (independently); i.e., restart g ($g = 1, ..., 12$) uses an initial $n \times k = 6 \times 2$ LHS design. After this initial stage (or iteration) $s = 0$, the next stages $s = 1, 2, ...$, T_g use independent simulation outputs with heterogeneous variances $\sigma_h^2(\boldsymbol{x}_i) = var[w_h(\boldsymbol{x}_i)]$. We can keep track of N (defined in (21)) per macroreplication in our algorithm, so we can allow OptQuest to run the same N.

Fig. 5 and Fig. 6 show $\widehat{\boldsymbol{x}}_{\min}$ for OptQuest in 35 macroreplications. Fig. 5 shows $\widehat{\boldsymbol{x}}_{\min}$ when we stop OptQuest manually as soon as the total number of simulation runs exceeds N of the corresponding

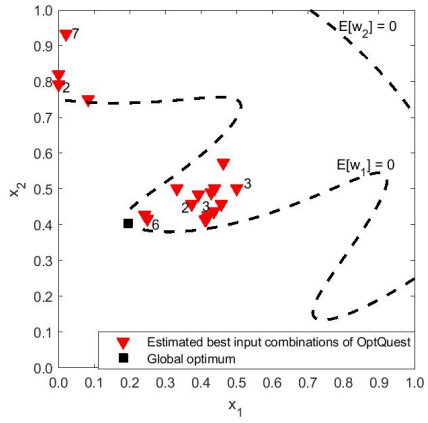

Figure 5: OptQuest's $\widehat{\boldsymbol{x}}_{\min}$ with the stopping criterion N; 35 macroreplications with $\alpha_{infe} = 10\%$ for the toy problem

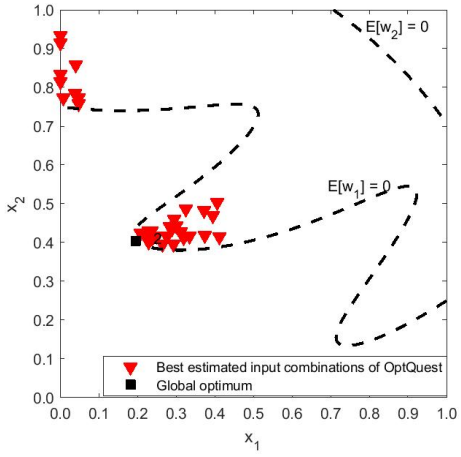

Figure 6: OptQuest's $\widehat{\boldsymbol{x}}_{\min}$ with the stopping criterion n; 35 macroreplications with $\alpha_{infe} = 10\%$ for the toy problem

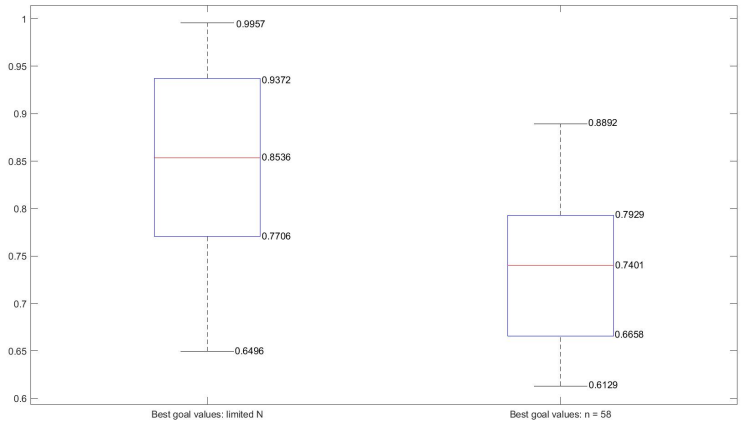

Figure 7: OptQuest's $\widehat{\boldsymbol{x}}_{\min}$ in 35 macroreplications with $\alpha_{infe} = 10\%$ for the toy problem: limited N on the left, and $n = 58$ on the right

macroreplication of our algorithm. Actually, several macroreplications give the same $\widehat{\boldsymbol{x}}_{\min}$ (e.g., $\widehat{\boldsymbol{x}}_{\min} = (0.022, 0.9337)'$ in seven macroreplications), as denoted by the numbers in this Figure. Whereas OptQuest gives no infeasible $\widehat{\boldsymbol{x}}_{\min}$-results, our algorithm gives 3 out-of 35 infeasible $\widehat{\boldsymbol{x}}_{\min}$-results (remember that 3/35 is controlled by $\alpha_{infe} = 10\%$ in our algorithm). Furthermore, this Figure shows that 11 (= 7 + 2+ 1 +1) of the 35 $\widehat{\boldsymbol{x}}_{\min}$-results are close to the local minimum $\boldsymbol{x}_{\mathrm{C}} = (0, 0.7)'$, and the remaining 24 (= 35 - 11) $\widehat{\boldsymbol{x}}_{\min}$-results are close to the global minimum $\boldsymbol{x}_{\mathrm{A}} \approx (0.1954, 0.4044)'$. Our algorithm, however, gives $\widehat{\boldsymbol{x}}_{\min}$-results close to $\boldsymbol{x}_{\mathrm{A}}$ in all 35 macroreplications; see again Fig. 2. Fig. 6 shows $\widehat{\boldsymbol{x}}_{\min}$-results when we follow the recommendation in OptQuest and set the total number of input combinations n between 58 and 65. Actually, even the smaller n (namely, 58) causes OptQuest to run longer than N does; yet, 9 of the 35 macroreplications give $\widehat{\boldsymbol{x}}_{\min}$-results near $\boldsymbol{x}_{\mathrm{C}}$ (and 26 $\widehat{\boldsymbol{x}}_{\min}$-results closer to $\boldsymbol{x}_{\mathrm{A}}$). We add that 2 macroreplications give the same $\widehat{\boldsymbol{x}}_{\min}$; namely, $\widehat{\boldsymbol{x}}_{\min} = (0.2500, 0.4167)'$ (the number 2 is not displayed in Fig. 6, because this plot has too many triangles close together).

Figure 7 displays the boxplots for $\widehat{y}_{0;\,\min}$, which correspond with $\widehat{\boldsymbol{x}}_{\min}$ in Fig. 5 and Fig. 6. The median values of $\widehat{y}_{0;\,\min}$ are 0.8536 and 0.7401, so both values are far from the true global optimal value 0.5998.

Finally, we obtain boxplots (not displayed) for the total number of simulation observations required in 35 macroreplications for OptQuest and our algorithm. The median value for OptQuest is 58,527, which is more than four times bigger than the median value 14,382 for our algorithm (see again Figure 4). The ranges of the two boxplots do not overlap. We conclude that our algorithm is more efficient (requires less elapsed time) than OptQuest, when applied to the toy example.

6 An (s, S) inventory simulation with a service-level constraint

As we have already discussed in Section 1, inventory simulations with a service-level constraint are of major practical relevance. Now we investigate an (s, S) inventory simulation with a service-level constraint that was originally investigated in Bashyam and Fu (1998). In this simulation, a company places an order (with its supplier) of size $S - I_p$ as soon as the *inventory position* I_p drops below the reorder level s—where I_p denotes the physical inventory plus outstanding orders minus "backorders", which are customer orders that cannot be delivered immediately because the physical inventory is too small (backordering implies that there are no lost sales), and p denotes the review period $p = 1, 2, ..., P$, where P is the end of the simulation run. Bashyam and Fu (1998) select P = 30,000, assuming this gives steady-state simulation outputs. (So, the order size is not a constant $Q = S$ - s.)

Furthermore, Bashyam and Fu (1998) allows random demand D and lead times L, so the orders that the company places are not necessarily received in the order in which they are placed. This *order crossing* complicates the mathematical analysis, so analysts use simulation. Estimating the *optimal* inventory-control limits s_o and S_o is difficult, as the vast literature on inventory management shows.

Finally, Bashyam and Fu (1998) assumes that the goal variable—w_0 in our symbols—is the average total inventory *costs* excluding (the hard to quantify) cost of out-of-stock (causing back orders). The *disservice percentage* w_1 is the ratio of (i) total demand during P that is not satisfied from the physical stock, and (ii) total demand during that period. We assume that this percentage should not exceed 10%, so $E(w_1) \leq 0.10$ (Bashyam and Fu (1998) also investigates 1% and 5%, and Kleijnen and Wan (2007) investigates only 10%). Obviously, w_0 and w_1 are negatively correlated. (Our definition of w_1 implies that the service percentage is $1 - w_1$.)

Kleijnen and Wan (2007) assumes that D is exponentially distributed with mean 100, L is integer-valued Poisson distributed with mean 6, holding cost h is 1, variable ordering cost is 1, and fixed ordering cost K is 36 (we need the symbols h and K below; Bashyam and Fu (1998) investigates more cases.)

Table 1: Initial 6×2 design and its estimated outputs, in macroreplication 1; * denotes $\overline{w}_1 > 0.10$

s	S	$\overline{w}_0$	$\overline{w}_1$
750.0	1166.7	486.7022	0.2932 *
1183.3	1766.7	928.6275	0.0426
616.7	1100.0	453.7042	0.3549 *
816.7	1366.7	603.4781	0.1883 *
783.3	833.3	327.8253	0.5096 *
1116.7	1500.0	761.0117	0.0691

We use the following *input constraints* for s and S. For s we select the lower bound $E(D) \times E(L) =$

$100 \times 6 = 600$, which ignores the randomness of D and L, so it is too low to ensure proper service. For the upper bound we (rather arbitrarily) select $2 \times 600 = 1{,}200$. For S we (rather arbitrarily) select the same lower bound as for s (namely, 600), and for the upper bound we select $3 \times 600 = 1{,}800$. So, initially, we did not apply the reparametrization with $x_2 = Q = S - s$ (so, we did not apply LHS), but derived an alternative procedure that gives the first two columns in Table 1. For the same initial design we compare the performance of OptQuest and our algorithm. When we do parameterize this inventory problem, then we may vary Q around the popular *economic order quantity* (EOQ), which is $[2 \times K \times E(D)/h]^{1/2} = 85$; see Bashyam and Fu (1998). (Obviously, this EOQ ignores the randomness of D and L.)

Note that we must avoid numerical problems—when computing the Gaussian correlation function (11) and the corresponding gradient (12) (which contain $\exp[-\theta_{h;j'}(x_{*;j'} - x_{i:j'})^2]$)—so we *normalize* (or standardize) the simulation inputs s and S such that $0 \leq x_j \leq 1$. The input constraints in the (s, S) problem are $2k = 4$ box constraints plus the extra constraint $s \leq S$ (or $x_1 \leq x_2$).

Next we determine the number of *replications* m_i, using Law's rule (26). It turns out that $m_0 = 10$ does not require any additional replications; i.e., the pilot sample is too big. Therefore we next select $m_0 = 2$, and find that m_i is very small; namely, 2, 3 or 4. We conclude that $P = 30{,}000$ gives very accurate estimates.

Running the simulation with the $n = 6$ initial input combinations of Table 1 and the m_i replications gives the outputs $\overline{w}_0$ and $\overline{w}_1$ in macroreplication 1; see the last two columns of this Table. Qualitative knowledge about (s, S) inventory systems implies that $\overline{w}_1$ (disservice fraction) decreases as s or S increases (because higher s or S keeps the system farther away from the "danger zone" where out-of-stocks occur). In this Table, we denote $\overline{w}_1 > 0.10$ by *; so, we estimate that only 2 of the 6 (s, S) combinations are feasible. Furthermore, this Table shows that the combination in the last row is the best initial solution; i.e., this combination gives an acceptable disservice and the lowest cost $\overline{w}_0$.

Kleijnen and Wan (2007) compares several algorithms for estimating s_o and S_o, including OptQuest. We compare only our algorithm and the OptQuest algorithm (as we do for the toy example). For our comparison we apply both algorithms to the same (s, S) simulation model (there are different interpretations of (s, S) systems—e.g., not all interpretations use our definition of I_p—and programming codes—we follow the coding in Law (2015, pp. 48–61)).

6.1 Our algorithm versus OptQuest in inventory example

Fig. 8 displays the estimated optimal input combinations $(\widehat{s}_o, \widehat{Q}_o)$ with $\widehat{Q}_o = \widehat{S}_o$ - $\widehat{s}_o$ in our (s, S) inventory simulation, in 22 macroreplications for our algorithm and OptQuest. We have 22 macroreplications because our OptQuest licence expired after six weeks, which enabled us to obtain 22 macroreplications for OptQuest

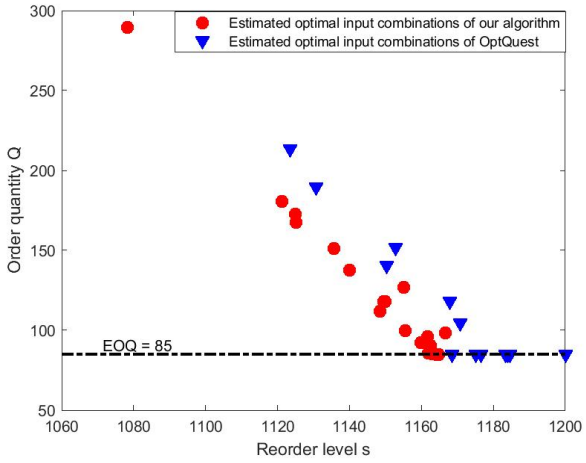

Figure 8: Estimated optimal input combination $(\widehat{s}_{\rm o}, \widehat{Q}_{\rm o})$ with $\widehat{Q}_{\rm o} = \widehat{S}_{\rm o}$ - $\widehat{s}_{\rm o}$ in (s, S) inventory simulation, in 22 macroreplications for our algorithm and OptQuest

applied to our inventory example (after 35 macroreplications for the toy example). This Figure also displays the EOQ, which implies that all 22 $(\widehat{s}_{\rm o}, \widehat{Q}_{\rm o})$-combinations—for our algorithm and OptQuest, respectively—give $\widehat{Q}_{\rm o} \geq {\rm EOQ} = 85$. Moreover, our algorithm gives combinations with smaller $\widehat{s}_{\rm o}$ and $\widehat{Q}_{\rm o}$ than OptQuest gives, which decreases the costs but might endanger the service. Therefore we next construct Fig. 9.

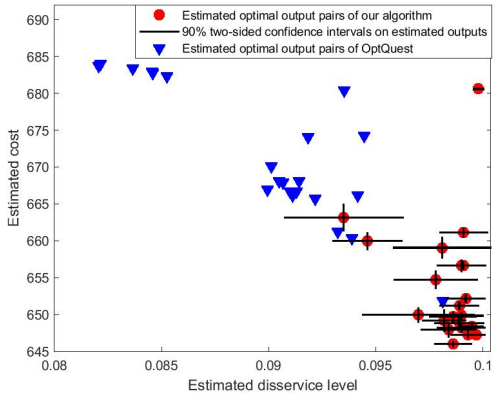

Figure 9: Scatterplot of estimated optimal disservice versus estimated optimal inventory cost, in 22 macroreplications; crosses denote 90% CIs

Fig. 9 displays a scatterplot of the estimated optimal disservice ($\bar{w}_1(\hat{\mathbf{x}}_{\rm min})$ in OptQuest, and $\hat{y}_1(\hat{\mathbf{x}}_{\rm min})$ in our algorithm) and the corresponding estimated optimal cost ($\bar{w}_{0;{\rm min}}$ and $\hat{y}_{0;{\rm min}}$) of our inventory simulation, in the same 22 macroreplications as in Fig. 8; the Figure also displays 90% CIs for each of these disservice and cost estimates. Projecting these 22 points for OptQuest and our algorithm onto the axes determines the boxplots. The median cost and disservice for our algorithm are 650 and 0.0990, and for OptQuest 668 and 0.0911. So, the percentage decrease in cost is 3%, while still satisfying the service-level threshold of 90%. In

practice, the cost saving is even higher when applying our algorithm instead of "experience" to optimize the (s, S) inventory system.

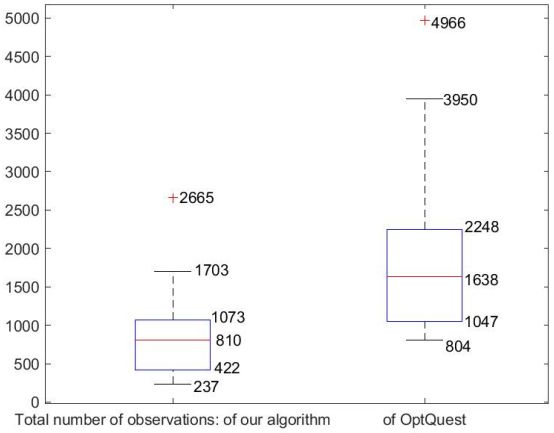

Figure 10: Boxplots of N (total number of simulation observations) of our algorithm (left) versus OptQuest (right) in 22 macroreplications of the inventory example

Fig. 10 displays boxplots of N (defined in (21) as a pseudo-measure for the elapsed times) of our algorithm and OptQuest. This Figure shows that the median N is 810 for our algorithm, and 1,638 for OptQuest; i.e., our algorithm requires much less time to get better estimates of the optimum.

7 Conclusions and future research

In this paper, we investigated constrained optimization problems in random simulation; i.e., we derived a novel algorithm, combining the KT conditions and EGO. More specifically, we account for both output constraints and input constraints, allow variance heterogeneity, and apply SK with a sample allocation rule. Our algorithm uses PS to estimate $\boldsymbol{x}$ that maximizes the current SK estimate of EGO's $\widehat{\text{MEI}}(\boldsymbol{x})$ multiplied by $\widetilde{\widetilde{\cos}}[\boldsymbol{x}]$, which quantifies how well the KT conditions hold at $\boldsymbol{x}$. Our algorithm uses multiple cores of our PC. Our algorithm turns out to be more efficient and effective than OptQuest is, in experiments with a mathematical example and an (s, S) inventory system.

Future research may investigate the following topics: (i)To improve the efficiency and effectiveness of our algorithm, we may customize PS or replace PS by a better optimizer. (ii) We may consider problems with $k \gg 2$ inputs. (iii) Besides continuous inputs we may investigate other types of input (e.g., integer inputs). (iii) Whereas we assume that the problem uses expected values, we may investigate probabilities such as $PR[w_{h'}(\boldsymbol{x}) > c_{h'}] \leq 0.10$; Kleijnen (2015, p. 222) gives references for Kriging with probabilities, and Chen et al. (2022) gives a more recent method and references.

Acknowledgement

We use the MATLAB code for stochastic Kriging that was developed by Sebastian Rojas Gonzalez (FWO Postdoctoral Fellow, Hasselt University and Ghent University, Belgium) while he was a Ph.D. student at KU Leuven (Belgium). To optimize our two examples through OptQuest (as an alternative for our own algorithm), we use a free 45-day trial version of OptQuest's SimWrapper (https://www.opttek.com/products/optquest/simwrapper); we thank Ben Thengvall (OptTek) for his help with OptQuest and its interface with the MATLAB code for our examples. We thank Inneke van Nieuwenhuyse (Hasselt University) and Wim van Beers for their help with several prior versions of our paper

References

Ankenman, B., B. Nelson, and J. Staum (2010) Stochastic kriging for simulation metamodeling. *Operations Research*, 58, no. 2, pp. 371–382

Bashyam, S. and M.C. Fu (1998) Optimization of (s, S) inventory systems with random lead times and a service-level constraint. *Management Science*, 44, pp.243–256

Boyd S. and L. Vandenberghe (2009), *Convex optimization; seventh printing with corrections*. Cambridge University Press, United Kingdom

Chen, J. ,M.M. Aral, S-H. Kim, C. Park, and Y. Xie (2022): Constrained Bayesian optimization and spatio-temporal surveillance for sensor network design in the presence of measurement errors. *Engineering Optimization*, pp. 1–16

Dykstra, R.L. (1970), Establishing the positive definiteness of the sample covariance matrix. *Annals of Mathematical Statistics*, 41, no.6, pp. 153–2154

Erickson, C.B., B.E. Ankenman, M. Plumlee, and S.M. Sanchez (2021), Gradient-based criteria for sequential experiment design. *Quality and Reliability Engineering*, pp. 3084-3107

Frazier, P.I. (2018), A tutorial on Bayesian optimization. https://arxiv.org/pdf/1807.02811.pdf

Gramacy, R.B., G.A. Gray, S. Le Digabel, H.K.H. Lee, P. Ranjan, G. Wells, and S.M. Wild (2016) Modeling an augmented Lagrangian for blackbox constrained optimization, *Technometrics*, 58, no. 1, pp. 1–11 (Comments by several authors, and rejoinder by Gramacy et al. pp. 12–29)

Guidetti, X., A. Rupenyan, L. Fassl, M.Nabavi, and J. Lygeros (2022), Advanced manufacturing configuration by sample-efficient batch Bayesian optimization. arXiv:2205.11827v1 [cs.LG] 24 May 2022

Jalali, H., I. van Nieuwenhuyse, and V. Picheny (2017), Comparison of Kriging-based algorithms for simulation optimization with heterogeneous noise. *European Journal of Operational Research*, 261, pp. 279–301

Jones, D., M. Schonlau, and W. Welch (1998). Efficient global optimization of expensive blackbox functions. *Journal of Global Optimization*, 13, pp. 455–492

Kleijnen, J.P.C. (2015), *Design and analysis of simulation experiments; second edition.* Springer

Kleijnen, J.P.C. and E. Mehdad (2014), Multivariate versus univariate Kriging metamodels for multi-response simulation models. *European Journal of Operational Research*, 236, pp. 573–582

Kleijnen, J. P. C. and W. van Beers (2022), Statistical tests for cross-validation of Kriging models. *INFORMS Journal on Computing*, 34, no. 1, pp. 607-621

Kleijnen J.P.C., W. van Beers, and I. van Nieuwenhuyse (2010) Constrained optimization in simulation: a novel approach. *European Journal of Operational Research*, 202, pp. 164–174

Kleijnen J.P.C., I. van Nieuwenhuyse, and W.C.M van Beers (2022), Constrained optimization in simulation: efficient global optimization and Karush-Kuhn-Tucker conditions. CentER Discussion Paper Series, No. 2022-015

Kleijnen, J. P. C. and J. Wan (2007), Optimization of simulated systems: OptQuest and alternatives. *Simulation Modelling Practice and Theory,* 15, pp. 354–362

Kolman, B. and D.R. Hill (2008), *Elementary Linear Algebra with Applications; 9th edition.* Pearson International Edition, Upper Saddle, New Jersey

Law, A.M. (2015), *Simulation modeling and analysis; fifth edition.* McGraw-Hill, Boston

L'Ecuyer, P. (2015), Random number generation with multiple streams for sequential and parallel computing. *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti

Liu, H., S. Xu, X. Chen, X. Wang, and Q. Ma (2017), Constrained global optimization via a DIRECT-type constraint-handling technique and an adaptive metamodeling strategy. *Structural and Multidisciplinary Optimization*, 55, no. 1, pp. 155–177

Loeppky, J.L., J. Sacks, and W. Welch (2009) Choosing the sample size of a computer experiment: a practical guide. *Technometrics*, 51, no. 4, pp. 366–376

Luo, J., L.J. Hong, B.L.. Nelson, and Y. Wu (2015), Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. *Operations Research,* 63, no. 5, pp. 1177–1194

Mahévas, S., V. Picheny, P. Lambert, N. Dumoulin, L. Rouan, J. Soulié, D. Brockhoff, S. Lehuta, R. Le Riche, R. Faivre, H. Drouineau (2019), A practical guide for conducting calibration and decision-making optimisation with complex ecological models. *Preprints*, 2019120249 (https://doi.org/10.20944/preprints201912.0249.v1)

Pandita, P., N. Awalgaonkar, I. Bilionis, and J. Panchal (2019), Learning arbitrary quantities of interest from expensive black-box functions through Bayesian sequential optimal design. arXiv:1912.07366v1 [stat.ML] 16 Dec 2019

Pearce, M.A.L. , M. Poloczek, and J. Branke (2022), Bayesian optimization allowing for common random numbers. *Operations Research*, preprint

Pedrielli, G., S. Wang, S.H. Ng (2020), An extended two stage sequential optimization approach: properties and performance. *European Journal of Operational Research*, 287, pp. 929–945

Pourmohamad, T. and H.K.H. Lee (2020), The statistical filter approach to constrained optimization. *Technometrics*, 62, no. 3, pp. 303–312

Pourmohamad T. and H.K.H. Lee (2021), Bayesian optimization via barrier functions. *Journal of Computational and Graphical Statistics*, accepted

Qing, J., H.B. Moss, T. Dhaene, and I. Couckuyt (2022), $\{PF\}^2ES$: Parallel feasible Pareto frontier entropy search for multi-objective Bayesian optimization under unknown constraints. arXiv preprint arXiv:2204.05411, 2022

Sóbester, A., S. J. Leary, and A. J. Keane (2005), On the design of optimization strategies based on global response surface approximation models. *Journal of Global Optimization*, 33, no. 1, pp. 31–59

Tao, I., G. Zhao, and S. Ren (2020), An efficient Kriging-based constrained optimization algorithm by global and local sampling in feasible region. *Journal of Mechanical Design*, 142, no. 5, pp. 051401-1 – 051401-15

Wang, Z. and M. Ierapetritou (2018), Constrained optimization of black-box stochastic systems using a novel feasibility enhanced Kriging-based method. *Computers and Chemical Engineering*, 118, pp. 210–223

Appendix 1: Common random numbers

In this appendix we investigate CRN with the usual assumption; namely, the number of replications per input combination is constant so $m(\boldsymbol{x}_i) = m$. We investigate two analysis methods; namely, SK and OK. CRN analyzed by SK requires the estimates $s[w_h(\boldsymbol{x}_i), w_h(\boldsymbol{x}_{i'})]$ with $i, i' = 1, ..., n$. These $s[w_h(\boldsymbol{x}_i), w_h(\boldsymbol{x}_{i'})]$ give the $n \times n$ matrix $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$, which is *nonsingular* only if $m > n$, as Dykstra (1970) proves. Our initial design uses $n = (k+1)(k+2)/2$ if $k \leq 6$ and $n = 5k$ if $k > 6$, following Tao et al. (2020)'s n for sequential optimization in deterministic simulation. Whichever n is selected, a high m may be required to obtain a non-singular $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$.

The literature on SK with CRN is very limited. Chen et al. (2012) does investigate SK with CRN, reporting a higher MSPE and a lower variance for the estimated gradient $\widehat{\nabla}(\boldsymbol{x})$. We focus on constrained optimization, and try to explain why this optimization implies that CRN do not perform well. Obviously, CRN with $m(\boldsymbol{x}_i) = m$ conflicts with the popular *allocation* rules that require more replications for input combinations with higher output variability (quantified by $var[w(\boldsymbol{x}_i)]$ or $var[w(\boldsymbol{x}_i)]^{1/2}$).

Note that Pearce et al. (2022) allows CRN, introduces a novel GP metamodel, and uses the Knowledge Gradient (KG)—in a Bayesian framework—to solve unconstrained optimization. We, however, use a GP that is conceptually simpler and requires the estimation of fewer parameters; moreover, we use MEI (which is simpler and therefore more popular than KG) in constrained (instead of unconstrained) optimization. (Pearce et al. (2022) also reviews the literature on unconstrained optimization with or without CRN.)

CRN with $m_i = m$ gives m IID vectors $\mathbf{w}_{h;r}$, each with the n components $w_{i;h;r}$ with mean $E(w_{i;h})$ and variance $\sigma^2_{i;h}$, while $w_{i;h}$ and $w_{i';h}$ have covariance $\sigma_{i;i';h}$. Like Chen et al. (2012), we estimate this $\sigma_{i;i';h}$ through

$$s(\overline{w}_{i;h}, \overline{w}_{i';h}) = \frac{\sum_{r=1}^{m}(w_{i;h;r} - \overline{w}_{i;h})(w_{i';h;r} - \overline{w}_{i';h})}{(m-1)m}. \tag{28}$$

Now we derive OK with CRN, which requires neither $s^2[w_h(\boldsymbol{x}_i)]$ nor $s[w_h(\boldsymbol{x}_i), w_h(\boldsymbol{x}_{i'})]$; moreover, $m > 1$ suffices. Originally, OK was derived for deterministic simulation, which implies that $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$ vanishes; i.e., the OK predictor (say) $\widehat{y}_{\mathrm{OK};h}$ and its estimated MSPE simplify drastically, compared with the SK predictor and its estimated MSPE. If $\widehat{\boldsymbol{\Sigma}}_{M;h}$ is *nearly singular* (or ill-conditioned), then we may add a small *nugget* effect; i.e., $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$ becomes $d_h \mathbf{I}_{n \times n}$ where d_h is a small number that is determined by the CPU accuracy; see Gramacy and Lee (2012), and Lophaven et al. (2002, p. 11). However, the original interpretation of the nugget effect is found in geostatistics; namely, the noise is determined by measurement errors etc.; this noise is denoted by d_h—or better σ^2_h; see Cressie (1993).

Now we introduce a new OK estimator for random simulation with CRN and $m_i = m$. We point out that $\mathbf{w}_{h;r}$ is the simulation output h in replication r with a fixed seed, so $\mathbf{w}_r$ is *not random* anymore (Pearce et

al. (2022) also emphasizes that the simulation output is deterministic, once the seed is fixed). Therefore, we apply OK to this $\mathbf{w}_{h;r}$ and obtain $\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x}|\boldsymbol{X},\boldsymbol{W}_{n\times t})$ or briefly $\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})$,which uses $\widehat{\boldsymbol{\psi}}(\boldsymbol{X},\boldsymbol{w_{h;r}})$:

$$\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x}) = \widehat{\mu}_{h;r} + \widehat{\boldsymbol{\sigma}}_{M;h;r}(\boldsymbol{x})'\widehat{\boldsymbol{\Sigma}}^{-1}_{M;h;r}(\mathbf{w}_{h;r} - \widehat{\mu}_{h;r}\mathbf{1}_n) \tag{29}$$

This $\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})$ would be the *best linear unbiased predictor* (BLUP) if the Kriging parameters $\boldsymbol{\psi}_h$ were known and the Kriging metamodel were perfectly valid so $E[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})] = E[w_h(\boldsymbol{x})]$. Furthermore, we assume that the (nonlinear) estimator $\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})$ is normally distributed. Then the best estimator of $E[w_h(\boldsymbol{x})]$ is the OK predictor averaged over the m replications:

$$\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x}) = \frac{\sum_{r=1}^{m} \widehat{y}_{\mathrm{OK};\,h;r}(\boldsymbol{x})}{m}. \tag{30}$$

The $\mathbf{w}_{h;r}$ give m IID $\widehat{y}_{\mathrm{OK};\,h;r}(\boldsymbol{x})$, so

$$var[\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})] = \frac{var[\widehat{y}_{\mathrm{OK};\,h}(\boldsymbol{x})]}{m}. \tag{31}$$

So, $m \uparrow \infty$ implies $var[\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})] \downarrow 0$; i.e., $\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})$ *converges* to $E[w_h(\boldsymbol{x})]$.

If the predictor is unbiased, then the MSPE reduces to the *extrinsic variance*; also see the subscript M in $\widehat{y}_h(\boldsymbol{x}_*)$ (obviously, $\widehat{y}_{\mathrm{OK};\,h}(\boldsymbol{x})$ is a special case of $\widehat{y}_h(\boldsymbol{x})$; i.e., $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$ vanishes in $\widehat{y}_h(\boldsymbol{x})$ and $s^2[\widehat{y}_h(\boldsymbol{x})]$). We follow the usual practice, and estimate this variance by the plug-in estimator

$$s^2_M[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})] = \widehat{\tau}^2_{h;r} - \widehat{\boldsymbol{\sigma}}_{M;h;r}(\boldsymbol{x})'\widehat{\boldsymbol{\Sigma}}^{-1}_{M;h;r}\widehat{\boldsymbol{\sigma}}_{M;h;r}(\boldsymbol{x}) + \frac{[1 - \mathbf{1}_n'\widehat{\boldsymbol{\Sigma}}^{-1}_{M;h;r}\widehat{\boldsymbol{\sigma}}_{M;h;r}(\boldsymbol{x})]^2}{\mathbf{1}_n'\widehat{\boldsymbol{\Sigma}}^{-1}_{M;h;r}\mathbf{1}_n}. \tag{32}$$

Though this variance estimator is not normally distributed, we simply use the average (instead of a robust estimator such as the sample median):

$$\overline{s^2_M}[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})] = \frac{\sum_{r=1}^{m} s^2_M[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})]}{m}. \tag{33}$$

Notice that (33) and (32) have the same expected value (namely, the extrinsic variance, given $\boldsymbol{X}$ and $\boldsymbol{W}_{n\times t}$), but (33) has a smaller variance:

$$var\{\overline{s^2_M}[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})]\} = \frac{var\{s^2_M[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})]\}}{m}.$$

So, $m \uparrow \infty$ implies $var\{\overline{s_M^2}[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})]\} \downarrow 0$; i.e., $\overline{s_M^2}[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})]$ *converges* to the true extrinsic variance.

It is well known that if $\boldsymbol{x} = \boldsymbol{x}_i$, then $s_M^2[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})]$ (defined in (32)) is exactly zero (OK is a perfect interpolator), for all r-values. Hence, its average $\overline{s_M^2}[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})]$ is also zero. However, if $\boldsymbol{x} = \boldsymbol{x}_i$, then there is still intrinsic noise; i.e., $\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x}_i) = w_{i;h;r}$ so

$$s^2[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x}_i)] = s^2(w_{i;h;r}) = \frac{\sum_{r=1}^m (w_{i;h;r} - \overline{w}_{i;h})^2}{(m-1)}.$$

This gives

$$s^2[\overline{\widehat{y}}_{\mathrm{OK};h;r}(\boldsymbol{x}_i)] = \frac{s^2[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x}_i)]}{m}.$$

For a general $\boldsymbol{x}$, we use

$$s_{\overline{e}}^2[\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})] = \frac{\sum_{r=1}^m [\widehat{y}_{\mathrm{OK};\ h;r}(\boldsymbol{x}) - \overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})]^2}{(m-1)m}; \tag{34}$$

also see the subscript $\overline{e}$ in $\widehat{y}_h(\boldsymbol{x}_*)$. So, $m \uparrow \infty$ gives $s_{\overline{e}}^2[\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})] \downarrow 0$; also see (31). Altogether, we decide to use

$$s^2[\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})] = \overline{s_M^2}[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})] + s_{\overline{e}}^2[\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})]. \tag{35}$$

We use the average of $\widehat{\nabla}[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})]$ (estimated gradient of $\widehat{y}_{\mathrm{OK};\ h}$ at $\boldsymbol{x}$, given $\boldsymbol{X}$ and $\boldsymbol{W}_{n\times t.r}$—for CRN with $r = 1, ..., m$):

$$\overline{\widehat{\nabla}}[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})] = \frac{\sum_{r=1}^m \widehat{\nabla}[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x})]}{m} \text{ with } h = 0, ..., t-1. \tag{36}$$

Note that our method is inspired by Kleijnen (2015, pp. 109–110)'s sensitivity analysis (not optimization) through *linear regression* analysis in random simulation with CRN and variance heterogeneity; i.e., instead of *generalized LS* (GLS) with estimated $\widehat{\boldsymbol{\Sigma}}_e$, this analysis may use *ordinary LS* (OLS). Furthermore, CRN may be combined with *antithetic random numbers* (ARN); e.g., Chih (2013) uses such a combination in a second-order polynomial metamodel.

Note that our algorithm uses only the point estimators $s^2[\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x})]$ and $\overline{\widehat{\nabla}}[\widehat{y}_{\mathrm{OK};h}(\boldsymbol{x})]$, and not their estimated variances.

Note that LOO-CV with CRN (analyzed through SK or OK) implies that $\overline{w}_{i;h}$ and $\widehat{y}_{i;h}$ or $\overline{\widehat{y}}_{\mathrm{OK};i;h}$(in the numerator) are not independent: $\widehat{y}_{i;h}$ or $\overline{\widehat{y}}_{\mathrm{OK};i;h}$ depend on $w_{i';h}$ $(i' \neq i)$, and $\overline{w}_{i;h}$ also depends on $w_{i';h}$. However, we ignore this dependence.

If our algorithm uses OK with CRN, then it replaces $\widehat{y}$ by $\overline{\widehat{y}}_{\mathrm{OK}}$.

The condition $m_i = m$ implies that our allocation rule does not apply. If PS revisits an old point, then we add one replication to each of the B points near the estimated boundary; if PS gives a new point, then

we add m replications to that point (so, $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$ remains well defined, in SK; OK does not use $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$). If $B \geq 5$, then we re-estimate all Kriging parameters; the condition $B \geq 5$ is meant to give acceptable MLEs of the parameters $\boldsymbol{\psi}_h$.

The next algorithm is a pseudo-code for CRN analyzed by SK. This algorithm closely follows the algorithm for IRN analyzed by SK, presented in the main text. However, the condition $m_i = m$ implies that the algorithm does not apply the sample-allocation rule. We delete all comments on the algorithm in the main text.

Algorithm 2

1. Select the algorithm's control variables α_{E}, α, α_{infe}, $\alpha_{\min}$, ε, m, n.
2. Sample the initial design matrix $\boldsymbol{X}$ with n combinations of the k simulation inputs x_j that satisfies the input constraints $f_g(\boldsymbol{x}) \leq c_g$ with $g = 1, ..., v$.
3. Use $\boldsymbol{X}$ as input for the simulation model, to obtain m replications using CRN, which gives $\mathbf{W}_{n\times t;r}$.
4. Apply LOO-CV to $(\boldsymbol{X}, \mathbf{W}_{n\times t;r})$.
5. If LOO-CV rejects the SK models, then obtain a new replication for all n combinations; replace m by $m+1$; return to Step 4.
6. Use $(\boldsymbol{X}, \mathbf{W}_{n\times t;r})$ to compute $\widehat{\boldsymbol{\psi}}_h$, $\widehat{\boldsymbol{\Sigma}}_{\overline{e};h}$, $\widehat{y}_h(\boldsymbol{x}_i)$, and $s^2[\widehat{y}_h(\boldsymbol{x}_i)]$.
7. Use $\widehat{y}_h(\boldsymbol{x}_i)$ to compute $\widehat{y}_{0;\,\min} = \min_i\, \widehat{y}_0(\boldsymbol{x}_i)$ if $\widehat{y}_{h'}(\boldsymbol{x}_i) + z_{(1-\alpha)}s[\widehat{y}_{h'}(\boldsymbol{x}_i)] \leq c_{h'}$ with $h' = 1, ..., t-1$ and $\alpha = \alpha_{\text{infe}}$; else $\widehat{y}_{0;\,\min} = \infty$.
8. Use $\widehat{y}_h(\boldsymbol{x}_i)$ to compute $\widehat{\nabla}_h[\widehat{y}_h(\boldsymbol{x}_i)]$.
9. Apply PS to estimate $\widehat{\boldsymbol{x}}_{\text{o}}$, which maximizes $\widehat{\text{MEI}}(\boldsymbol{x}) \times \widetilde{\cos}[\boldsymbol{x}]$ such that $\boldsymbol{x}$ satisfies the $t-1$ CIs for the constrained outputs h' and the v input constraints.
10. If PS finds a feasible point $\widehat{\boldsymbol{x}}_{\text{o}}$ that is an old point, then add one replication to all B points near the estimated boundary; else if PS finds a feasible point $\widehat{\boldsymbol{x}}_{\text{o}}$ that is a new point, then add m replications to this point; return to Step 6.
11. If PS does not find a feasible point $\widehat{\boldsymbol{x}}_{\text{o}}$, then replace α by $\alpha/2$. If $\alpha \geq \alpha_{\min}$, then return to Step 9; else if $\alpha < \alpha_{\min}$ and $\widehat{\text{MEI}}(\boldsymbol{x}) < \varepsilon \times |\widehat{y}_{0;\,\min}|$, then go to the final Step.
12. Present the estimated optimal goal output $\widehat{y}_{0;\,\min}$.

The next algorithm uses OK to analyze CRN with $m_i = m$ replications.

Algorithm 3

1. Select the algorithm's control variables α_{E}, α, α_{infe}, $\alpha_{\min}$, ε, m, n.
2. Sample the initial design matrix $\boldsymbol{X}_{n\times k}$ with n combinations of the k simulation inputs x_j that satisfies the input constraints $f_g(\boldsymbol{x}) \leq c_g$ with $g = 1, ..., v$.
3. Use $\boldsymbol{X}_{n\times k}$ as input for the simulation model, to obtain m replications using CRN, which gives $\mathbf{W}_{n\times t;r}$.
4. Apply LOO-CV to $(\boldsymbol{X}_{n\times k}, \mathbf{W}_{n\times t;r})$.
5. If LOO-CV rejects the OK models, then obtain a new replication for all n combinations; replace m by $m+1$; return to Step 4.
6. Use $(\boldsymbol{X}_{n\times k}, \mathbf{W}_{n\times t;r})$ to compute $\widehat{\boldsymbol{\psi}}_{h;r}$, $\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x}_i)$, $s^2[\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x}_i)]$.
7. Use $\overline{\widehat{y}}_{\mathrm{OK};h}(\boldsymbol{x}_i)$ to compute $\overline{\widehat{y}}_{\mathrm{OK};0;\ \min} = \min_i \overline{\widehat{y}}_{\mathrm{OK};0}(\boldsymbol{x}_i)$ if $\overline{\widehat{y}}_{\mathrm{OK};h'}(\widehat{\boldsymbol{x}}_i) + z_{(1-\alpha)} s[\overline{\widehat{y}}_{\mathrm{OK};h'}(\widehat{\boldsymbol{x}}_i)] \leq c_{h'}$ with $h' = 1, ..., t-1$ and $\alpha = \alpha_{\mathrm{infe}}$; else $\overline{\widehat{y}}_{\mathrm{OK};0;\ \min} = \infty$.
8. Use $\widehat{y}_{\mathrm{OK};h;r}(\boldsymbol{x}_i)$ to compute $\widehat{\nabla}[\widehat{y}_{\mathrm{OK};\ h;r}(\boldsymbol{x}_i)]$ and its average $\overline{\widehat{\nabla}}[\widehat{y}_{\mathrm{OK};\ h}(\boldsymbol{x}_i)]$
9. Apply PS to estimate $\widehat{\boldsymbol{x}}_{\mathrm{o}}$, which maximizes $\widehat{\mathrm{MEI}}(\boldsymbol{x}) \times \widetilde{\widehat{\cos}}[\boldsymbol{x}]$ such that $\boldsymbol{x}$ satisfies the $t-1$ CIs for the constrained outputs h' and the v input constraints.
10. If PS finds a feasible point $\widehat{\boldsymbol{x}}_{\mathrm{o}}$ that is an old point, then add one replication to all B points near the estimated boundary; else if PS finds a feasible point $\widehat{\boldsymbol{x}}_{\mathrm{o}}$ that is a new point, then add m replications to this point; return to Step 6.
11. If PS does not find a feasible point $\widehat{\boldsymbol{x}}_{\mathrm{o}}$, then replace α by $\alpha/2$. If $\alpha \geq \alpha_{\min}$, then return to Step 9; else if $\alpha < \alpha_{\min}$ and $\widehat{\mathrm{MEI}}(\boldsymbol{x}) < \varepsilon \times |\overline{\widehat{y}}_{\mathrm{OK};0;\ \min}|$, then go to the final Step.
12. Present the estimated optimal goal output $\overline{\widehat{y}}_{\mathrm{OK};0;\ \min}$.

Now we discuss CRN in the *toy example*. We assume that CRN indeed "work"; i.e., CRN create *positive* correlations between $e_h(\boldsymbol{x})$ and $e_h(\boldsymbol{x'})$ (outputs of the same type h, at points $\boldsymbol{x}$ and $\boldsymbol{x}'$). Furthermore, we assume that $\rho_{i;i';h}$ increases, as the simulated input combinations $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$ resemble each other more; i.e., we use the Gaussian correlation function, which uses $(x_{i;j} - x_{i';j})^2$ (so, our assumption resembles the assumption for the *extrinsic* noise in a GP). Initially we used LHS with $n = 20$ (instead of 6) midpoints. Combining this LHS and the Gaussian correlation function implies that the toy example has $\min_i (x_{i;j} - x_{i';j})^2$

$= (1/20)^2 = 0.0025$ and $\max_i(x_{i;j} - x_{i';j})^2 = 0.950^2 = 0.9025$. Obviously, if $(x_{i;j} - x_{i';j}) \downarrow 0$, then $\rho_{i;i';h} \uparrow 1$; if $(x_{i;j} - x_{i';j}) \uparrow \infty$, then $\rho_{i;i';h} \downarrow 0$. However, $n = 20$ implies $\max_i(x_{i;j} - x_{i';j}) = 0.950$ ($\ll \infty$); we decide which value $\rho_{i;i';h}$ should have at $x_{i';j} = 0.975$ (biggest midpoint). Obviously, this value—together with $\rho_{i;i';h} = 1$ at $x_{i;j} = 0$—determines θ_j, which features in the Gaussian correlation function. Furthermore, we define a $\rho_{i;i';h}$ between 0.5 and 1.0 as *high*, and between 0.0 and 0.5 as *low*. To select specific values for $\rho_{i;i';h}$ in these intervals, we use uniform sampling. Furthermore, we assume that $\theta_1 = \theta_2$, which is perfect for e_0 and e_2; for simplicity's sake, we also assume $\theta_1 = \theta_2$ for e_1. For example, $\rho_{i;i';h} = 0.5$ gives $\theta_1 = \theta_2 = 0.3128$, and $\rho_{i;i';h} = 0.1$ gives $\theta_1 = \theta_2 = 2.0781$.

Altogether we obtain *many* covariance matrixes; namely, one matrix per output type, for small noise and big noise, per initial 20×20 design $\boldsymbol{X}$, which is sampled per macroreplication and increases as new points are added. To illustrate these matrixes, we display the upper-left 5×5 submatrix of the 20×20 (symmetric) covariance matrix for e_1—for small noise and high CRN effect—with $\sigma_{e;1}(\boldsymbol{x})$ (instead of $\sigma^2_{e;1}(\boldsymbol{x})$) on the main diagonal, and the (scale-free) $\rho_{i;i';1}$ (so $\text{cov}[e_1(\boldsymbol{x}_i), e_1(\boldsymbol{x}_{i'})] = \rho_{i;i';1} \times \sigma_{e;1}(\boldsymbol{x}_i) \times \sigma_{e;1}(\boldsymbol{x}_{i'})$:

$$\begin{bmatrix} 1.21 & 0.97 & 0.94 & 0.94 & 0.98 \\ & 1.32 & 0.93 & 0.91 & 0.93 \\ & & 1.18 & 0.99 & 0.87 \\ & & & 0.81 & 0.87 \\ & & & & 0.76 \end{bmatrix} \tag{37}$$

The values $\rho_{1;2;1} = 0.97$ and $\rho_{1;3;1} = 0.94$ in this matrix imply that $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are closer than $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ are. We must check whether $\boldsymbol{\Sigma}_{e;20\times 20}$ is *positive definite* (PD). MATLAB offers two methods: (i) "attempt Cholesky factorization", and (ii) "check eigenvalues"; we use method (ii).

We assume *zero cross-correlations* $\rho_{i;0;1}$, $\rho_{i;0;2}$, and $\rho_{i;1;2}$ in the toy example, whereas a (random) *discrete-event* simulation model with multiple output types (so, $t > 1$) gives non-zero cross-correlations because the t outputs are generated by the same simulation model (also see the inventory example in Section 6). Unfortunately, we discovered that the toy example with non-zero cross-correlations gives serious numerical problems (we must then use a 60×60 covariance matrix that uses 20 3×3 submatrixes $\boldsymbol{\Sigma_e}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ instead of the preceding three 20×20 matrixes).

Once we have confirmed that $\boldsymbol{\Sigma}_{h;n\times n}$—with initial value $n = 20$—is PD, we sample the n-variate $\boldsymbol{e}_{h;n}$ from the *multi-variate normal density* (MND) with all n means equal to zero and the non-diagonal covariance matrix $\boldsymbol{\Sigma}_{h;n\times n}$:

$$\boldsymbol{e}_{h;n} \sim \text{N}_n(\boldsymbol{0}_n, \boldsymbol{\Sigma}_{h;n\times n}) \tag{38}$$

where our toy example initially uses $n = 10k = 20$, so $\boldsymbol{\Sigma}_{h;n\times n}$ is 20×20 and $h = 0, 1, 2$; we call the corresponding $\boldsymbol{X}$ the *initial* or *pilot* design. To sample from a MND, we use the MATLAB function *mvnrnd*. Given $\boldsymbol{e}_{h;n}$ sampled through (38), we compute the corresponding w_h ($h = 0,1, 2$).

Next, our algorithm finds an estimated optimal point $\widehat{\boldsymbol{x}}_{\rm o}$. There are two possibilities: (i) This $\widehat{\boldsymbol{x}}_{\rm o}$ is an *old* point, so a replication is added for both this point and all other old points "close" to the boundary, using CRN; i.e., m increases by 1 and the augmented I/O simulation data becomes $(\boldsymbol{X}, w_{h;r}(\widehat{\boldsymbol{x}}_{{\rm o};r}))$ (with $h = 0$, 1, 2 and $r = 1, ..., m$) where $w_{h;r}(\widehat{\boldsymbol{x}}_{{\rm o};r})$ uses $\boldsymbol{e}_{h;n}$ sampled through (38). (ii) This $\widehat{\boldsymbol{x}}_{\rm o}$ is a *new* point (near the boundary), so m replications are obtained for this point using the same PRNs as the old points used for the corresponding replications; i.e., m does not increase but n does increase by 1 and the augmented I/O simulation data becomes $(\boldsymbol{X}, w_{h;r}(\widehat{\boldsymbol{x}}_{\rm o}))$ where $w_{h;r}(\widehat{\boldsymbol{x}}_{{\rm o};r})$ is sampled through (40)—explained next. If $\widehat{\boldsymbol{x}}_{\rm o}$ is "very close" to one of the old points, then DACE gives the error message "Multiple design sites are not allowed". In our toy example (with $0 \le x_j \le 1$) we consider $\widehat{\boldsymbol{x}}_{\rm o}$ to be an old point if the absolute values of the differences in both input dimensions are less than 0.001 (so m becomes $m+1$). Note that (by definition) each *iteration* after the initial design increases either m or n (i.e., adds either a column or a row to $\boldsymbol{W}_h$).

If $\widehat{\boldsymbol{x}}_{\rm o}$ is a new point, then $\widehat{\boldsymbol{x}}_{\rm o}$ should give outputs that are correlated with the outputs of the old outputs (because of CRN); e.g., if CRN happens to give relatively high old outputs (i.e., many sampled values of $\boldsymbol{e}_{h;n}$ exceed $\boldsymbol{0}_n$), then the new outputs should also be relatively high (i.e., $e_{h;n+1}(\widehat{\boldsymbol{x}}_{\rm o}) > 0$). We define $\boldsymbol{e}_{h;n+1}$ $= (\boldsymbol{e}_{h;n}, e_{h;{\rm o}})'$ where $e_{h;{\rm o}} = e_h(\widehat{\boldsymbol{x}}_{\rm o})$. Obviously, $E(\boldsymbol{e}_{h;n}) = \boldsymbol{0}_n$ and $\boldsymbol{\Sigma}_{h;n\times n}$ is the $n \times n$ covariance matrix in (38). Furthermore, $\sigma^2_{{\rm o};h}$ denotes the variance of $e_{{\rm o};h}$ at $\widehat{\boldsymbol{x}}_{\rm o}$. Finally, $\boldsymbol{\sigma}_{n;{\rm o}}$ denotes the n-dimensional vector with the covariances between the n old intrinsic noises and the new intrinsic noise at $\widehat{\boldsymbol{x}}_{\rm o}$, for response h. Altogether, we define the following $(n+1) \times (n+1)$ covariance matrix

$$\boldsymbol{\Sigma}_{h;(n+1)\times(n+1)} = \begin{bmatrix} \boldsymbol{\Sigma}_{h;n\times n} & \boldsymbol{\sigma}_{n;{\rm o}} \\ \boldsymbol{\sigma}'_{n;{\rm o}} & \sigma^2_{{\rm o};h} \end{bmatrix}. \tag{39}$$

Obviously, this matrix should be PD. If the sampled values for $\rho_{i;{\rm o};h}$—together with the n old points $\boldsymbol{x}_i$ (i $= 1, ..., n$) and the new point $\boldsymbol{x}_{\rm o}$—do not give a $\boldsymbol{\sigma}_{n;{\rm o}}$ that gives a PD matrix, then we resample $\rho_{i;{\rm o};h}$ and compute the corresponding $\theta_1 = \theta_2$—until the matrix is PD. To sample from the *conditional* distribution of $e_h(\widehat{\boldsymbol{x}}_{\rm o})$ given $\boldsymbol{e}_n$, we change the mean vector and the covariance matrix in (38) (e.g., Mittelhammer (1996)):

$$\boldsymbol{e}_h(\widehat{\boldsymbol{x}}_{\rm o}) \mid \boldsymbol{e}_{h;n} \sim {\rm N}(\boldsymbol{\sigma}'_{n;{\rm o}}\boldsymbol{\Sigma}^{-1}_{h;n\times n}\boldsymbol{e}_{h;n}, \sigma^2_{{\rm o};h} - \boldsymbol{\sigma}'_{n;{\rm o}}\boldsymbol{\Sigma}^{-1}_{h;n\times n}\boldsymbol{\sigma}_{n;{\rm o}}) \tag{40}$$

We have already seen that $\boldsymbol{\Sigma}_{h;n\times n}$ is PD (see (37)), but now (40) uses $\boldsymbol{\Sigma}^{-1}_{h;n\times n}$—which might be *nearly*

singular (MATLAB then gives the error message: "the matrix is close to be singular" and we add a nugget effect). Fortunately, our toy example does not give these numerical problems.

Each *macroreplication* samples a new initial LHS design, independent of the designs in the other macroreplications. This new design implies a new (20×20) $\mathbf{\Sigma}_{h;n\times n}$.

Appendix 2: The heterogeneous variances of the intrinsic noise in the toy problem

Jalali et al. (2017) gives the following two equations:

$$\begin{aligned}[\max_{\boldsymbol{x}}\ w_h(\boldsymbol{x}) + b_h] \times a_h &= \max_{\boldsymbol{x}}\ \sigma_{e;h}(\boldsymbol{x}) \\ [\min_{\boldsymbol{x}}\ w_h(\boldsymbol{x}) + b_h] \times a_h &= \min_{\boldsymbol{x}}\ \sigma_{e;h}(\boldsymbol{x}),\end{aligned} \tag{41}$$

and defines the *range* of $w_h(x)$ as $R_h = \max_{\boldsymbol{x}}\ E[w_h(\boldsymbol{x})] - \min_{\boldsymbol{x}}\ E[w_h(x)]$ and distinguishes between small noise $0.6R_h$ or $0.15R_h$ and big noise $6.0R_h$ or $1.5R_h$. Because that publication estimates the global minimum of an unconstrained problem, the *best* case means that $\sigma_{e;h}(\boldsymbol{x})$ decreases linearly as $w_h(\boldsymbol{x})$ decreases. So, the right-hand sides of (41) use

$$\begin{aligned}\max_{\boldsymbol{x}}\ \sigma_{e;h}(\boldsymbol{x}) &= 0.60R_h \text{ or } 6.00R_h \\ \min_{\boldsymbol{x}}\ \sigma_{e;h}(\boldsymbol{x}) &= 0.15R_h \text{ or } 1.50R_h.\end{aligned} \tag{42}$$

Combining (41) and (42), we can determine a_h and b_h in (41); this gives the desired linear equation

$$\sigma_{e;h}(\boldsymbol{x}) = a_h b_h + a_h E[w_h(\boldsymbol{x})]. \tag{43}$$

The *worst* case means that $\sigma_{e;h}(\boldsymbol{x})$ increases linearly as $E[w_h(\boldsymbol{x})]$ decreases. Combining the analogue of (41) with (42) gives

$$\begin{aligned}\{b_h + \min_{x}\ E[w_h(\boldsymbol{x})]\} \times a_h &= \max_{\boldsymbol{x}}\ \sigma_{e;h}(\boldsymbol{x}) \\ \{b_h + \max_{x}\ E[w_h(\boldsymbol{x})]\} \times a_h &= \min_{\boldsymbol{x}}\ \sigma_{e;h}(\boldsymbol{x}).\end{aligned} \tag{44}$$

Using the data in the toy example, we compute $\min_x\ E[w_0(\boldsymbol{x})] = 0$ and $\max_x\ E[w_0(\boldsymbol{x})] = 2$ so $R_0 = 2$;

hence, (43) (best case with smallest noise) gives

$$\sigma_{e;0}(\boldsymbol{x}) = 0.30 + 0.45E[w_0(\boldsymbol{x})]. \tag{45}$$

Furthermore, $\min_x\ E[w_1(\boldsymbol{x})]$ = -1.5 and $\max_x\ E[w_1(\boldsymbol{x})]$ = 1.6711 so R_1 = 3.1711, and $\min_x\ E[w_2(\boldsymbol{x})]$ = -1.5 and $\max_x\ E[w_2(\boldsymbol{x})]$ = 0.5 so R_2 = 2. The main text gives $\sigma_{e;1}(\boldsymbol{x})$ and $\sigma_{e;2}(\boldsymbol{x})$ for the best case with smallest noise; furthermore, $\sigma_{e;h}(\boldsymbol{x})$ for the worst case with biggest noise is

$$\begin{aligned}
\sigma_{e;0}(\boldsymbol{x}) &= 12.0 - 4.5E[w_0(\boldsymbol{x})] \\
\sigma_{e;1}(\boldsymbol{x}) &= 12.2766 - 4.5E[w_1(\boldsymbol{x})] \\
\sigma_{e;2}(\boldsymbol{x}) &= 5.25 - 4.5E[w_2(\boldsymbol{x})]
\end{aligned} \tag{46}$$

Appendix 3: EI versus MEI

Jones et al. (1998) derives the following estimator of EI where $w_{\min} = \min_{1\le i\le n}\ w(x_i)$ and we use our symbol $\widehat{y}_{\mathrm{OK}}(\boldsymbol{x})$:

$$\begin{aligned}
\widehat{\mathrm{EI}}(\boldsymbol{x}) = (w_{\min} - \widehat{y}_{\mathrm{OK}}(\boldsymbol{x}))\,\Phi\left(\frac{w_{\min} - \widehat{y}_{\mathrm{OK}}(\boldsymbol{x})}{s[\widehat{y}_{\mathrm{OK}}(\boldsymbol{x})]}\right) \\
+ s[\widehat{y}_{\mathrm{OK}}(\boldsymbol{x})]\phi\left(\frac{w_{\min} - \widehat{y}_{\mathrm{OK}}(\boldsymbol{x})}{s[\widehat{y}_{\mathrm{OK}}(\boldsymbol{x})]}\right).
\end{aligned} \tag{47}$$

Inspired by Jalali et al. (2017), we define the following estimator of MEI where $\boldsymbol{x}_{\min}$ is the point that gives $w_{\min}$, $\widehat{y}(\boldsymbol{x}_{\min})$ replaces $w_{\min}$:

$$\begin{aligned}
\widehat{\mathrm{MEI}}(\boldsymbol{x}) = (\widehat{y}(\boldsymbol{x}_{\min}) - \widehat{y}(\boldsymbol{x}))\,\Phi\left(\frac{\widehat{y}(\boldsymbol{x}_{\min}) - \widehat{y}(\boldsymbol{x})}{s[\widehat{y}(\boldsymbol{x})]}\right) \\
+ s[\widehat{y}(\boldsymbol{x})]\phi\left(\frac{\widehat{y}(\boldsymbol{x}_{\min}) - \widehat{y}(\boldsymbol{x})}{s[\widehat{y}(\boldsymbol{x})]}\right).
\end{aligned} \tag{48}$$

Appendix 4: Details for toy problem

This Appendix shows how our algorithm proceeds through various stages (or iterations)—before it reaches its final estimated optimum.

Fig. 11 shows dashed curves to denote the *white-box* constraints $E(w_{h'}) = 0$ ($h' = 1,\ 2$) (which are the analogues of the solid curves for the deterministic constraints). These curves—combined with the (non-displayed) goal function $E[w_0(\boldsymbol{x})] = x_1 + x_2$—give the (black) square that denotes $\boldsymbol{x}_{\mathrm{A}} \approx (0.1954, 0.4044)'$.

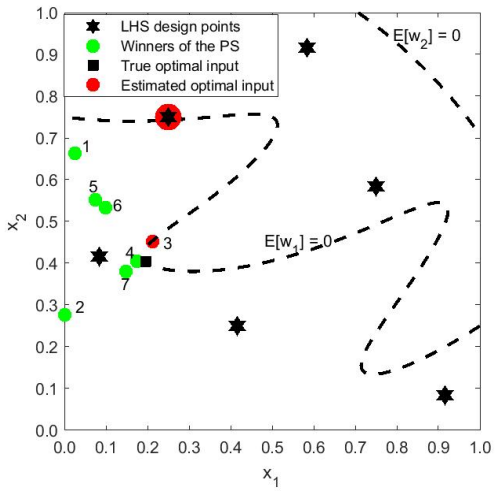

Figure 11: Toy example: input combinations in initial design and next iterations, for a restart ending near A in macroreplication 57 with $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, and 20 PS starting points

Each *restart* begins in iteration 0 using a new initial LHS design. We discuss two of the twelve restarts, in macroreplication 57 (there are 60 macroreplications); namely, one restart that ends near $\boldsymbol{x}_{\text{A}}$ and has relatively few iterations after iteration 0 (namely, seven so our discussion is relatively short), and one restart that ends near $\boldsymbol{x}_{\text{C}} = (0.00, 0.75)$ and has relatively many iterations including revisits of old input combinations. All restarts and macroreplications use different LHS designs, but use the same α_{infe} (namely, 10%), $m_{\min}$ (= 10), and number of PS starting points (20).

Table 2: Iterations of a restart ending near point A in macroreplication 57

Iter.	$\hat{\mathbf{x}}_{\min}$	$\hat{y}_{0;\min}$	$E[w_0(\hat{\mathbf{x}}_{\min})]$	$\hat{\mathbf{x}}_{\text{o}}$	$\widehat{MEI}$	$\widehat{\widetilde{cos}}$
0	(0.2500, 0.7500)	0.9106	1.0000	(0.0250, 0.6625)	0.3712	0.9980
1	(0.2500, 0.7500)	0.9935	1.0000	(0.0006, 0.2769)	0.7276	1.0000
2	(0.2500, 0.7500)	0.9699	1.0000	(0.2125, 0.4515)	0.4198	0.9993
3	(0.2500, 0.7500)	0.9778	1.0000	(0.1740, 0.4047)	0.4632	0.9996
4	(0.2125, 0.4515)	0.5680	0.6640	(0.0750, 0.5515)	0.0138	0.6307
5	(0.2125, 0.4515)	0.5777	0.6640	(0.0989, 0.5325)	0.0125	0.3748
6	(0.2125, 0.4515)	0.5726	0.6640	(0.1490, 0.3797)	0.1033	0.9989
7	(0.2125, 0.4515)	0.6068	0.6640			

The (black) stars denote the initial $n = 6$ input combinations selected through LHS with midpoints. The preprocessing (with LOO-CV) obtains the. required number of replications for these combinations. The star within a (red) ball denotes $\hat{\mathbf{x}}_{\min} = (0.2500, 0.7500)'$, in iteration 0; this $\hat{\mathbf{x}}_{\min}$ is also displayed in Table 2, in the row for iteration—or "Iter."—0 (because we know the true I/O functions, we know that this $\hat{\mathbf{x}}_{\min}$ is indeed feasible). The algorithm uses PS, to compute $\widehat{\boldsymbol{x}}_{\text{o}}$ that maximizes the product of $\widehat{\text{MEI}}(\boldsymbol{x})$ and $\widehat{\widetilde{\text{cos}}}(\boldsymbol{x})$;

see Table 2, last three columns.

In the Figure, the (green) solid balls denote $\widehat{\boldsymbol{x}}_{\rm o}$ in iteration 1, ..., 7. The dark (red instead of light green) ball denotes the final estimate of $\hat{\mathbf{x}}_{\rm min} = (0.2125, 0.4515)'$, which equals $\widehat{\boldsymbol{x}}_{\rm o}$ obtained in iteration 2.

Furthermore, this Table shows that iteration 0 gives $\hat{y}_{0;\rm min} = 0.9106$. We (not the algorithm) also know the corresponding $E[w_0(\hat{\mathbf{x}}_{\rm min})] = 0.2500 + 0.7500 = 1.0000$. The winner of the PS is $\widehat{\boldsymbol{x}}_{\rm o} = (0.0250, 0.6625)'$. This $\widehat{\boldsymbol{x}}_{\rm o}$ maximizes the product of $\widehat{\rm MEI}(\boldsymbol{x})$ and $\widetilde{\widehat{\cos}}(\boldsymbol{x})$; actually, $\widehat{\rm MEI}(\widehat{\boldsymbol{x}}_{\rm o}) = 0.3712$ (which is a sizable improvement of $\hat{y}_{0;\rm min} = 0.9106$) and $\widetilde{\widehat{\cos}}(\boldsymbol{x}) = 0.9980$ (which is close to the maximum value 1). At the end of an iteration, the algorithm runs the simulation model with $\widehat{\boldsymbol{x}}_{\rm o}$ as input, obtains (not displayed) $w_{h;r}(\widehat{\boldsymbol{x}}_{\rm o})$, and uses the augmented I/O data to re-estimate (not displayed) $\widehat{y}_h$, $s[\widehat{y}_h]$, and $\widehat{\nabla}_h$. Iteration 1 uses these $\widehat{y}_h$ and $s[\widehat{y}_h]$, and estimates that $\widehat{\boldsymbol{x}}_{\rm o} = (0.0250, 0.6625)'$ (result of iteration 0) is infeasible; so, $\hat{\mathbf{x}}_{\rm min}$ remains the same as in iteration 0. However, $\hat{y}_{0;\rm min}$ becomes 0.9935. Of course, if $\hat{\mathbf{x}}_{\rm min}$ remains the same, then $E[w_0(\hat{\mathbf{x}}_{\rm min})]$ remains the same; namely, 1.0000. The re-estimated $\widehat{y}_h$, $s[\widehat{y}_h]$, and $\widehat{\nabla}_h$ give a new $\widehat{\boldsymbol{x}}_{\rm o}$ =$(0.0006, 0.2769)'$. The algorithm obtains $w_{h;r}(\widehat{\boldsymbol{x}}_{\rm o})$, and re-estimates $\widehat{y}_h$, $s[\widehat{y}_h]$, and $\widehat{\nabla}_h$.

Iteration 2 estimates $\widehat{\boldsymbol{x}}_{\rm o} = (0.0006, 0.2769)'$ to be infeasible; so, $\hat{\mathbf{x}}_{\rm min}$ does not change. The new $\widehat{\boldsymbol{x}}_{\rm o}$ becomes $(0.2125, 0.4515)'$.

Iteration 3 estimates this $\widehat{\boldsymbol{x}}_{\rm o}$ to be infeasible, so this $\widehat{\boldsymbol{x}}_{\rm o}$ does not become the new $\hat{\mathbf{x}}_{\rm min}$. Actually, the algorithm evaluates all old simulated combinations, and finds that $\hat{\mathbf{x}}_{\rm min}$ remains the same as in iteration 0. The new $\widehat{\boldsymbol{x}}_{\rm o}$ becomes $(0.1740, 0.4047)'$.

Iteration 4 re-estimates $\hat{y}_h$ and $s[\hat{y}_h]$, and estimates $\widehat{\boldsymbol{x}}_{\rm o} = (0.1740, 0.4047)'$ to be infeasible. Moreover, the new $\hat{\mathbf{x}}_{\rm min}$ becomes an old $\hat{\mathbf{x}}_{\rm o}$; namely, $(0.2125, 0.4515)'$ (see iteration 2). And so the algorithm proceeds.

Iteration 7 gives the final $\hat{\mathbf{x}}_{\rm min} = (0.2125, 0.4515)'$ with $\hat{y}_{0;\rm min} = 0.6068$ (iteration 4 has already given the same $\hat{\mathbf{x}}_{\rm min}$, but a different $\hat{y}_{0;\rm min} = 0.5680$). The algorithm stops because $\alpha = 0.0063$ falls below its lower bound 0.01.

Fig. 12 displays $\hat{y}_{0;\rm min}$ as a function of N_q for restart q that requires preprocessing with 237 runs (so the average number of replications per combination is $237/6 = 39.5$) and the total N is 483. The total number of replications ($\sum_{i=1}^{n} m_{i;s;q}$) in iteration $s = 7$ is much smaller than in iteration 1 (the distance between two consecutive balls projected onto the horizontal axis is smaller in iteration 7). This Figure also shows + signs that denote symmetric 90% CIs around $\widehat{y}_{0;\ \rm min}$. Obviously, the lengths of these CIs tend to decrease as N increases. (We do not display the Figure for $E[w_0(\hat{\mathbf{x}}_{\rm min})]$, because that Figure resembles Fig. 12.)

Fig. 13 is the analogue of Fig. 11, but finishes near C instead of A.

Fig. 14 is the analogue of Fig. 12. It shows that this restart requires many more simulation runs in the preprocessing; namely, 1,497. The final value of N is 9,883. Its final solution is farther away from the true global optimum.

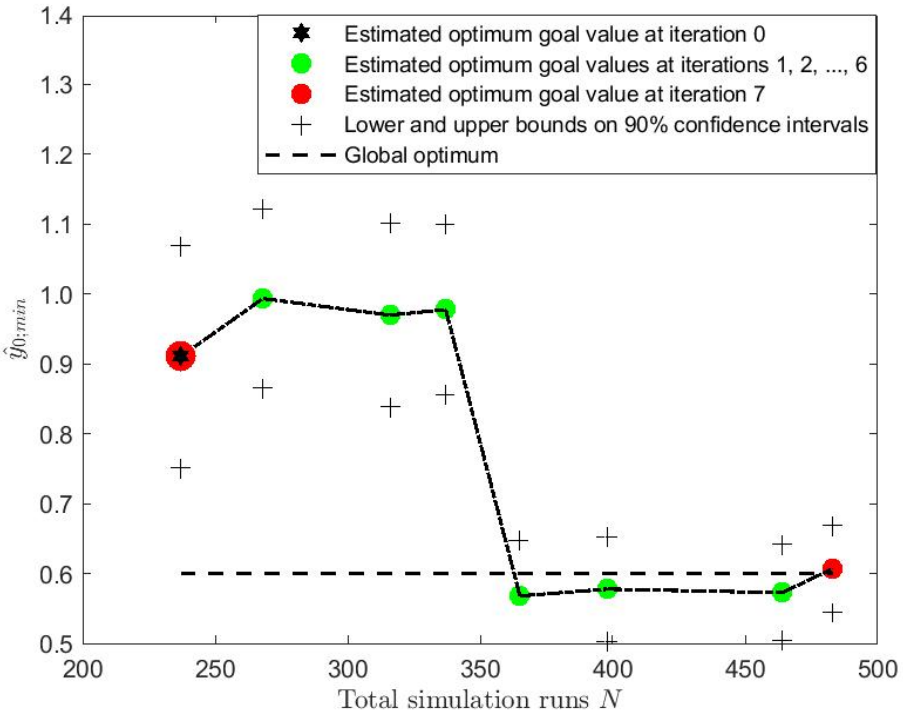

Figure 12: Toy example: estimated optimal goal value when simulation outputs are added in iterations 0 through 7, for a restart ending near A in macroreplication 57 with $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, and 20 PS starting points

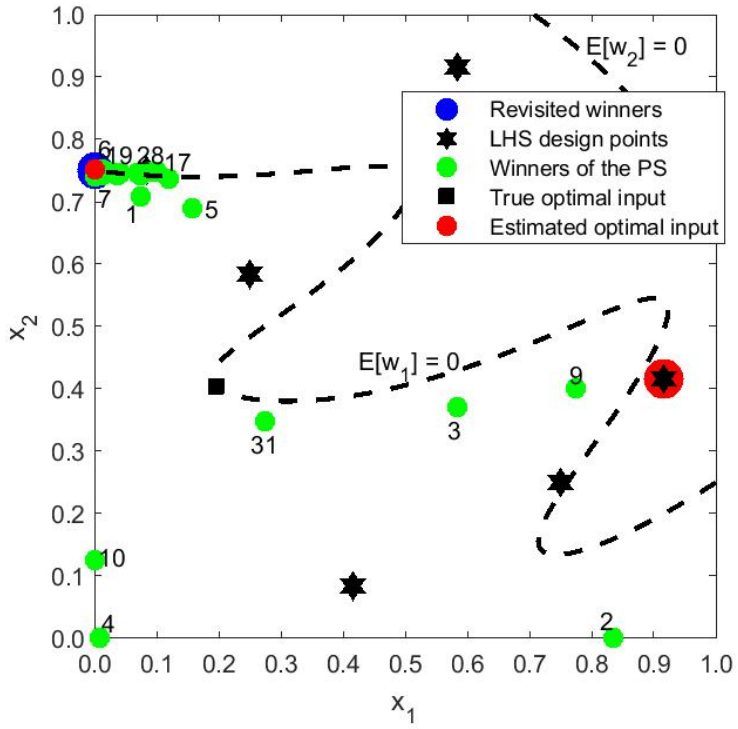

Figure 13: Toy example: input combinations in initial design and next iterations, for a restart ending near C in macroreplication 57 with $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, and 20 PS starting points

References

Chen, X., B. Ankenman, and B.L. Nelson (2012), The effects of common random numbers on stochastic kriging metamodels. *ACM Transactions on Modeling and Computer Simulation*, 22, no. 2, Article 7

Chih M (2013), A more accurate second-order polynomial metamodel using a pseudo-random number assignment strategy. *Journal of the Operational Research Society*, 64:198–207

Cressie, N.A.C. (1993), *Statistics for spatial data. (Revised edition).* Wiley, New York

Dykstra, R.L. (1970), Establishing the positive definiteness of the sample covariance matrix. *Annals of Mathematical Statistics*, 41, no.6, pp. 153–2154

Gramacy, R.B. and H.K.H. Lee (2012), Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22, no 3, pp. 713–722

Jalali, H., I. v an Nieuwenhuyse, and V. Picheny (2017), Comparison of Kriging-based algorithms for simulation optimization with heterogeneous noise. *European Journal of Operational Research*, 261, pp. 279–301

Jones, D., M. Schonlau, and W. Welch (1998). Efficient global optimization of expensive blackbox functions. *Journal of Global Optimization*, 13, pp. 455–492

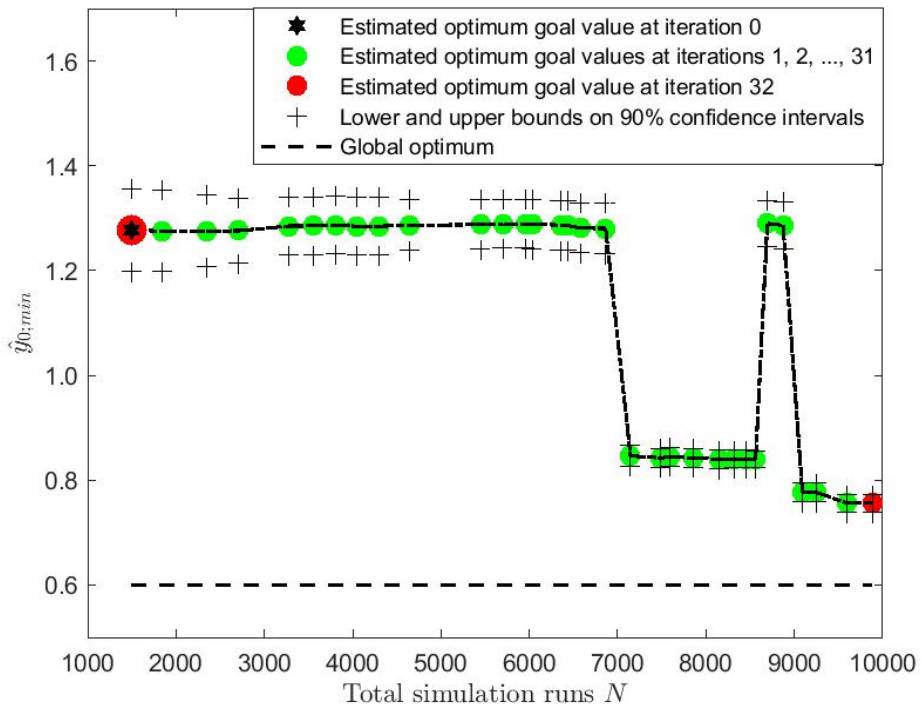

Figure 14: Toy example: estimated optimal goal value when simulation outputs are added in iterations 0, 1, ..., for a restart ending near C in macroreplication 57 with $\alpha_{\text{infe}} = 10\%$, $m_{\min} = 10$, and 20 PS starting points

Kleijnen, J.P.C. (2015), *Design and analysis of simulation experiments; second edition.* Springer

Lophaven S.N., H.B. Nielsen, J. Sondergaard (2002), DACE: A Matlab Kriging Toolbox, Version 2.0 IMM Technical University of Denmark, Kongens Lyngby, Denmark

Mittelhammer R.C. (1996). Mathematical Statistics for Economics and Business. Springer: New York

Pearce, M.A.L. , M. Poloczek, and J. Branke (2022), Bayesian optimization allowing for common random numbers. *Operations Research*, preprint

Tao, I., G. Zhao, and S. Ren (2020), An efficient Kriging-based constrained optimization algorithm by global and local sampling in feasible region. *Journal of Mechanical Design*, 142, no. 5, pp. 051401-1 – 051401-15