Made available by Hasselt University Library in https://documentserver.uhasselt.be

Constrained optimization in simulation: efficient global optimization and Karush-Kuhn-Tucker conditions Peer-reviewed author version

Kleijnen, Jack P. C.; Angun, Ebru; VAN NIEUWENHUYSE, Inneke & van Beers, Wim C. M. (2024) Constrained optimization in simulation: efficient global optimization and Karush-Kuhn-Tucker conditions. In: Journal of Global Optimization,.

DOI: 10.1007/s10898-024-01448-3 Handle: http://hdl.handle.net/1942/44890

Constrained optimization in simulation:

efficient global optimization and Karush-Kuhn-Tucker conditions

Jack P.C. Kleijnen¹, Ebru Angün², Inneke van Nieuwenhuyse³, and Wim C.M. van Beers⁴

¹ Department of Management, Tilburg University (TiU), Tilburg, Netherlands

E-mail: kleijnen@tilburguniversity.edu, Orcid iD: https://orcid.org/0000-0001-8413-2366

² Industrial Engineering Department, Galatasaray University, Istanbul, Turkey

E-mail: eangun@gsu.edu.tr, Orcid iD: https://orcid.org/0000-0002-8199-9746

³ FlandersMake@UHasselt and Data Science Institute, Hasselt University, Hasselt, Belgium inneke.vannieuwenhuyse@uhasselt.be, Orcid iD: https://orcid.org/0000-0003-2759-3726 ⁴ Vlijmen, Netherlands

Abstract

We develop a novel methodology for solving constrained optimization problems in deterministic simulation. In these problems, the goal (or objective) output is to be minimized, subject to one or more constraints for the other outputs and for the inputs. Our methododology combines the "Karush-Kuhn-Tucker" (KKT) conditions with "efficient global optimization" (EGO). These KKT conditions are well-known first-order necessary optimality conditions in white-box mathematical optimization, but our method is the first EGO method that uses these conditions. EGO is a popular type of algorithm that is closely related to "Bayesian optimization" and "active machine learning", as they all use Gaussian processes or Kriging to approximate the input/output behavior of black-box models. We numerically compare the performance of our KKT-EGO algorithm and two alternative EGO algorithms, in several popular examples. In some examples our algorithm converges faster to the true optimum, so our algorithm may provide a suitable alternative.

Keywords: Karush-Kuhn-Tucker conditions, efficient global optimization, Bayesian optimization, machine learning, Kriging, Gaussian process

1 Introduction

In this paper we try to solve constrained optimization problems in simulation. We focus on the search for *global* optima of non-convex, multimodal problems. Furthermore, we focus on problems with at least one *binding* (or active) constraint at the optima, but we also discuss problems without such constraints.

To solve these problems, we combine EGO and the KKT conditions. The name EGO is proposed in [1], which refers to its predecessors [2] and [3]; also see [4], p. 42. EGO is closely related to *Bayesian optimization* (BO), and is also related to *active machine learning* (AML). Nowadays, the name BO is more popular than EGO; see the surveys [5] – [9]. We prefer the name BO if the methodology uses prior distributions and posterior distributions; nevertheless, for example, [10] uses the name BO, without explicit mentioning of these distributions. We follow the *frequentist* approach instead of the Bayesian approach. EGO, BO, and AML use *Kriging* —named after Danie Krige—or *Gaussian process* (GP) models, to analyze *input/output* (I/O) data. For brevity's sake, below we will use the name "EGO" instead of "EGO and/or BO" if confusion is not likely.

EGO methods are *heuristics*; the term "heuristic" has many meanings, but we define it as a problemsolving technique that produces approximately correct solutions. To the best of our knowledge, no EGO algorithm for optimization with output constraints has *guaranteed convergence* to the global optima ([11] provides a theoretical discussion of an algorithm for optimization over a compact domain that uses a statistical model of the goal function; however, [11] does not consider output constraints).

Currently, there are many variants of EGO, but none uses the KKT conditions. Nevertheless, these conditions are well-known first-order necessary optimality conditions in mathematical optimization (MO) or mathematical programming; see the seminal textbook [12] and the recent textbook [13]. However, MO assumes white-box problems; i.e., the output variables are known explicit functions of the input variables. EGO assumes black-box problems; i.e., the simulation outputs are unknown implicit functions—defined by the simulation model—of the simulation inputs. Consequently, we observe only a limited number of I/O simulation combinations. So, we do not know whether the simulation model has a convex I/O transfer function, and the simulation model itself does not give gradient information (the Kriging metamodel does, as we shall see in Section 4.2).

Each EGO variant uses its own *acquisition function*—or *infill criterion*—to select the point to be simulated in the next iteration. For our EGO variant we develop a novel acquisition function that multiplies the popular *expected improvement* (EI) by a specific *cosine* function (related to the popular coefficient of determination R^2) that quantifies how well the KKT conditions hold in case of a binding constraint. If there is no binding constraint, then the infill criterion multiplies EI by a function that quantifies how close the goal gradient is to the zero vector.

Our methodology is *flexible* because it is *modular*, so other authors can replace our specific modules (or building blocks) by their own preferred modules. Actually, our EGO methodology uses Kriging models that need to be differentiable at least once, as we shall see in Section 4.1. We shall discuss our modules and potential alternative modules, in several sections.

We numerically compare the performance of our algorithm with two *alternative* EGO algorithms that are defined in [14] and [10]. We think that these alternatives are representative of state-of-the-art EGO variants for constrained optimization in deterministic simulation; also see our literature review in the next section.

We apply these three algorithms to several popular mathematical examples and mechanical-engineering examples; namely, [15]'s mathematical "toy" example (which is also used in [10] and [16]), [17]'s constrained Hartmann-6 example, the tension-compression spring problem, and the three-bar truss design problem; Appendix 4 also details [17]'s Hartmann-6 example without a binding constraint. (The literature offers many more examples; e.g., [18] defines thirteen mathematical examples and four mechanical-engineering examples.)

We organize the rest of this paper as follows. Section 2 reviews selected publications on EGO. Section 3 formalizes constrained optimization and first-order optimality conditions, including a least squares model and a specific cosine function. Section 4 summarizes Kriging. Section 5 discusses the initial design (also known as the pilot design or training set). Section 6 details our novel EGO methodology. Section 7 presents the numerical results of our experiments with three EGO methods and five examples. Section 8 presents conclusions, including future research topics.

2 Literature review of EGO

Originally, [1] developed EGO for unconstrained optimization in deterministic simulation. Since then, EGO methods have become popular in computationally expensive black-box optimization—with or without constraints. Paraphrasing [10], p. 75, we state that EGO's Kriging metamodels have desirable properties: they closely approximate most simulation I/O functions (say) $w(\mathbf{x})$ where \mathbf{x} denotes an input combination, are much faster to compute than $w(\mathbf{x})$ is (because $w(\mathbf{x})$ is computed via the corresponding simulation model), and quantify the uncertainty of the Kriging predictor $\hat{y}(\mathbf{x})$ through the estimated standard deviation $s[\hat{y}(\mathbf{x})]$.

In the preceding section, we have already referred to the surveys [6]-[9]. Furthermore, [4] also reviews EI extensions for various types of optimization, including multiple-step (non-myopic) algorithms; [19] derives a two-step lookahead BO algorithm for constrained optimization. However, we focus on publications that impact our methodology and present a few basic formulas. Therefore, we do not discuss methods that use

trust regions, or Kriging but not EGO, or radial basis functions without EGO, etc. We present EGO methods for unconstrained optimization in Section 2.1 and for constrained optimization in Section $2.2.^{1}$

2.1 Unconstrained EGO

Section 1 implies that the essence of a specific EGO variant is its *acquisition function* (say) $a(\mathbf{x})$. In unconstrained optimization, the most popular infill criteria are the EI and the *probability of improvement* (PI). The following estimator of this PI is derived in [21] where Φ denotes the standard Gaussian (or normal) cumulative distribution function, and w_{\min} denotes the best simulated goal output among the *n* "old" (already simulated) outputs so $w_{\min} = \min_{1 \le i \le n} [w(\mathbf{x}_i)]$:

$$\widehat{\mathrm{PI}}(\boldsymbol{x}) = \Phi\left(\frac{w_{\min} - \widehat{y}(\boldsymbol{x})}{s[\widehat{y}(\boldsymbol{x})]}\right).$$
(1)

The estimator of EI is also derived in [21] where ϕ denotes the standard Gaussian density function:

$$\widehat{\mathrm{EI}}(\boldsymbol{x}) = (w_{\min} - \widehat{y}(\boldsymbol{x})) \Phi\left(\frac{w_{\min} - \widehat{y}(\boldsymbol{x})}{s[\widehat{y}(\boldsymbol{x})]}\right) + s[\widehat{y}(\boldsymbol{x})]\phi\left(\frac{w_{\min} - \widehat{y}(\boldsymbol{x})}{s[\widehat{y}(\boldsymbol{x})]}\right).$$
(2)

2.2 Constrained EGO

Following [10], we distinguish two types of EGO methods for constrained optimization: (i) *Classical* methods that combine EI or PI for the goal output (say) w_0 and the *probability of feasibility* (PF) for the constrained outputs $w_{h'}$ with h' = 1, ..., t - 1; see Section 2.2.1 below. (ii) *Hybrid* methods that combine EI or PI for w_0 and MO concepts for $w_{h'}$; see Section 2.2.2.

2.2.1 Classical constrained EGO

For constrained optimization, the literature replaces w_{\min} in (1) and (2) by

$$w_{0;\min} = \min_{1 \le i \le n} [w_0(\boldsymbol{x}_i): w_{h'}(\boldsymbol{x}_i) \le c_{h'}, \forall h'],$$
(3)

and $\hat{y}(\boldsymbol{x})$ by $\hat{y}_0(\boldsymbol{x})$; these replacements give $\widehat{\mathrm{Pl}}_0(\boldsymbol{x})$ and $\widehat{\mathrm{El}}_0(\boldsymbol{x})$. However, (3) assumes that the *n* old simulated points include at least one feasible point. Actually, in our numerical experiments we investigate two engineering problems with feasible areas so small that the initial designs may give no feasible points. We shall present a simple solution in Subsection 5.3.

¹ [20] reviews EI extensions for constrained optimization and other types of optimization. That review covers publications appearing in the recent twenty years. The review has 181 references—but does not refer to the publications on constrained optimization that we reference, except for [15].

Furthermore, the literature defines the estimator of the probability of feasibility for constraint h' (PF_{h'}) analogously to (1)):

$$\widehat{\mathrm{PF}}_{h'}(oldsymbol{x}) = \Phi\left(rac{c_{h'} - \widehat{y}_{h'}(oldsymbol{x})}{s[\widehat{y}_{h'}(oldsymbol{x})]}
ight)$$

These $\widehat{\mathrm{PF}}_{h'}(\boldsymbol{x})$ give the estimator of PF:

$$\widehat{\mathrm{PF}} = \Pi_{h'=1}^{t-1} \widehat{\mathrm{PF}}_{h'}.$$
(4)

Obviously, this $\widehat{\mathrm{PF}}$ treats all constraints as statistically independent.

 $\widehat{EI}_0(\boldsymbol{x})$ and \widehat{PF} are combined in [22]'s constrained EI (CEI) acquisition function:

$$a_{\rm CEI}(\boldsymbol{x}) = \widehat{\rm EI}_0(\boldsymbol{x}) \times \widehat{\rm PF}.$$
(5)

Clearly, the maximization of $a_{\text{CEI}}(\boldsymbol{x})$ implies unconstrained optimization. Carpio et al. [14] replaces $\widehat{\text{EI}}_0$ in (5) by $\widehat{\text{PI}}_0$, which gives

$$a_{\rm PI-PF}(\boldsymbol{x}) = \widehat{\rm PI}_0(\boldsymbol{x}) \times \widehat{\rm PF}.$$
(6)

We shall numerically evaluate (6), as a recent representative of classical constrained EGO.

2.2.2 Hybrid constrained EGO

Pourmohamad and Lee [10] derives barrier function (BF) methods, which outperform the (older) statistical filter (SF) methods in [16] and the augmented Lagrangian (AL) methods in [15]. These BF, SF and AL methods are hybrid; more precisely, BF methods—also known as MO's interior point methods—try to minimize $w_0(\mathbf{x})$ while ensuring that the boundary of the feasible region is never crossed. Therefore, BF methods add an extra term in the acquisition function that is a penalty for approaching this boundary; i.e., [10], eq. 18 uses

$$a_{\rm BF}(\boldsymbol{x}) = \widehat{\rm EI}_0(\boldsymbol{x}) + s^2[\widehat{y}_0(\boldsymbol{x})] \sum_{h'=1}^{t-1} (\log(-\widehat{y}_{h'}(\boldsymbol{x})) + \frac{s^2[\widehat{y}_{h'}(\boldsymbol{x})]}{2[\widehat{y}_{h'}(\boldsymbol{x})]^2})$$
(7)

where $-\hat{y}_{h'}(\boldsymbol{x}) > 0$. Like (5) and (6), maximization of $a_{BF}(\boldsymbol{x})$ implies unconstrained optimization. We note that [10] uses univariate Kriging (like we do; see Section 4.1), whereas its predecessor [16] uses multivariate Kriging.

3 Constrained optimization and Karush-Kuhn-Tucker conditions

To present the mathematical definition of the optimization problem with output constraints, we use the following symbols. We denote the multiple simulation outputs by $w_h(\mathbf{x})$ with h = 0, 1, ..., t - 1 and t > 1. The goal output to be minimized is w_0 ; the (t - 1) outputs $w_{h'}(\mathbf{x})$ (with h' = 1, ..., t - 1) are constrained outputs with the prespecified upper thresholds $c_{h'}$, so $w_{h'}(\mathbf{x}) \leq c_{h'}$. We call \mathbf{x} infeasible if $\exists h' : w_{h'}(\mathbf{x}) > c_{h'}$; an infeasible \mathbf{x} does not necessarily make the simulation model crash. We assume k continuous simulation inputs x_j (j = 1, ..., k). These x_j define the point or input combination $\mathbf{x} = (x_1, ..., x_k)'$. Whereas most publications assume that these x_j must satisfy the box constraints $l_j \leq x_j \leq u_j$, we allow the more general input constraints $f_g(\mathbf{x}) \leq c_g$ with g = 1, ..., v (e.g., $x_1 - x_2 \leq 0$). Altogether, we focus on the following type of constrained optimization problem (we use the compact notation that is also used in [10]):

$$\min_{\boldsymbol{x}} [w_0(\boldsymbol{x}): w_{h'}(\boldsymbol{x}) \le c_{h'}, \forall h', f_g(\boldsymbol{x}) \le c_g, \forall g].$$
(8)

Some publications (e.g., [10]) use $c_{h'} = 0$; i.e., they move $c_{h'}$ to the left-hand side in (8). Actually, [10]'s $a_{\rm BF}(\boldsymbol{x})$ (defined in (7)) requires $\hat{y}_{h'}(\boldsymbol{x}) < 0$, so the problem is defined such that $c_{h'} = 0$.

The KKT conditions use gradients. We denote the gradient of $w_h(\boldsymbol{x})$ at a new (not yet simulated) point \boldsymbol{x}_* by $\nabla_h(\boldsymbol{x}_*)$ with the k partial derivatives $\partial w_h/\partial x_j$ evaluated at \boldsymbol{x}_* . So, we assume that $w_h(\boldsymbol{x})$ is differentiable, which is a realistic assumption in many applications. A constraint is binding at \boldsymbol{x}_* if \boldsymbol{x}_* lies on the boundary (or frontier) of the feasible area; i.e., if \boldsymbol{x}_* changes \leq in (8) into = for at least one constraint so the *slack* of this constraint becomes exactly zero. Let h'' denote the *indexes* of the binding output constraints at \boldsymbol{x}_* and $A_\lambda(\boldsymbol{x}_*)$ the *index set* with the indexes h'' (obviously, $A_\lambda(\boldsymbol{x}_*)$ is empty if there are no binding output constraints at \boldsymbol{x}_*).

The input constraints $f_g(\boldsymbol{x}) \leq c_g$ may include the box constraints $x_j \leq u_j$ and $-x_j \leq -l_j$ ((8) requires that all constraints use \leq). These constraints imply that ∇_g is a vector with either 1 or -1 at position j and 0 at the (k-1) remaining positions with g = 1, ..., 2k. We let $A_{\mu}(\boldsymbol{x}_*)$ denote the *index set* with the indexes g' of the *binding input* constraints at \boldsymbol{x}_* .

These gradients give the *KKT stationarity conditions* (see [13], p. 274, [12], p. 243) where we denote the *Lagrangian* multiplier for binding output constraint h'' by $\lambda_{h''}$ and for binding input constraint g' by $\mu_{g'}$:

$$-\nabla_{0}(\boldsymbol{x}_{*}) = \sum_{h'' \in A_{\lambda}(\boldsymbol{x}_{*})} \lambda_{h''}(\boldsymbol{x}_{*}) \nabla_{h''}(\boldsymbol{x}_{*}) + \sum_{g' \in A_{\mu}(\boldsymbol{x}_{*})} \mu_{g'}(\boldsymbol{x}_{*}) \nabla_{g'}(\boldsymbol{x}_{*})$$

with $\lambda_{h''}(\boldsymbol{x}_{*}) \ge 0$ and $\mu_{g'}(\boldsymbol{x}_{*}) \ge 0.$ (9)

Obviously, (9) is an orthogonal projection of $-\nabla_0$ onto the linear space spanned by $\nabla_{h''}$, $\forall h''$ and $\nabla_{g'}$, $\forall g'$. If no constraint is binding, then elementary analysis gives the following first-order necessary optimality condition:

$$\nabla_0(\boldsymbol{x}_*) = \boldsymbol{0}.\tag{10}$$

We do not discuss *second-order* optimality conditions—for problems with or without binding constraints because our algorithm does not need these conditions, as we shall see in Section 6.

In general, an orthogonal projection can be interpreted as a least squares (LS) model; see [13], pp. 83–84. In our case, we compute $\lambda_{h''}$ and $\mu_{g'}$ in (9) via the LS model with the explained (dependent) variable $-\nabla_0$ and the explanatory (independent) variables $\nabla_{h''}(\boldsymbol{x}_*)$ and $\nabla_{g'}(\boldsymbol{x}_*)$. This LS computation gives $\lambda_{h''}$ and $\tilde{\mu}_{g'}$ where a tilde denotes LS—which give $-\widetilde{\nabla}_0$. If we knew $\nabla_{h''}(\boldsymbol{x}_*)$ and $\nabla_{g'}(\boldsymbol{x}_*)$ (in white-box optimization, we do know $\nabla_{h''}(\boldsymbol{x}_*)$ and $\nabla_{g'}(\boldsymbol{x}_*)$), then we could simply check whether the KKT conditions hold *exactly*, at a given point x_* ; i.e., whether $-\nabla_0(x_*)$ (the LS estimate of the left-hand side of (9)) perfectly coincides with the linear combination—with weights $\lambda_{h''}$ and $\mu_{q'}$ —of $\nabla_{h''}$ and $\nabla_{q'}(\boldsymbol{x}_*)$ —in the right-hand side. Perfect fit implies a zero angle between $-\nabla_0(\boldsymbol{x}_*)$ and $-\widetilde{\nabla}_0(\boldsymbol{x}_*)$; i.e. $\cos[-\nabla_0(\boldsymbol{x}_*), -\widetilde{\nabla}_0(\boldsymbol{x}_*)] = 1$ (this cos is related to R^2 with $0 \le R^2 \le 1$, which is a popular measure for fit in multiple regression, and is related to the correlation coefficient ρ with $-1 \leq \rho \leq 1$ in simple regression; perfect fit implies $R^2 = 1$ and in case of a single binding constraint $\rho = 1$). We abbreviate $\cos[-\nabla_0(\boldsymbol{x}_*), -\widetilde{\nabla}_0(\boldsymbol{x}_*)]$ to $\widetilde{\cos}(\boldsymbol{x}_*)$. If the fit is *imperfect*, then there may still be an *acute angle* between $-\nabla_0(\boldsymbol{x}_*)$ and $-\widetilde{\nabla}_0(\boldsymbol{x}_*)$; i.e., these two vectors point into the "same" direction so $0 < \widetilde{\cos}(x_*) \le 1$. If the angle is *obtuse* (i.e., the fit is very poor), then $-1 \le \widetilde{\cos}(x_*) < \infty$ 0. Obviously, $\cos[-\nabla_0(\boldsymbol{x}_*), -\widetilde{\nabla}_0(\boldsymbol{x}_*)] = \cos[\nabla_0(\boldsymbol{x}_*), \widetilde{\nabla}_0(\boldsymbol{x}_*)]$, so we use the following well-known formula (see, e.g., [23], p. 350) where \bullet denotes the inner product of two vectors and $||\nabla||$ denotes the L2 norm of the vector ∇ :

$$\widetilde{\cos}(\boldsymbol{x}_*) = \frac{\nabla_0(\boldsymbol{x}_*) \bullet \widetilde{\nabla}_0(\boldsymbol{x}_*)}{||\nabla_0(\boldsymbol{x}_*)|| \times ||\widetilde{\nabla}_0(\boldsymbol{x}_*)||}.$$
(11)

We code the LS computations in MATLAB; see the LS formulas in Appendix 1. However, other authors may use their own linear-regression code.

We illustrate the KKT conditions via Fig. 1, which is based on [15]'s example that we shall further discuss in Section 7.1. The dotted area denotes the feasible area, which lies between $w_1(x) = 0$ (a nonconvex function) and $w_2(x) = 0$. The point A is the global minimum. B and C are local minima; C has one binding input constraint, besides one binding output constraint. D and E are local maxima. F does not satisfy the KKT conditions. Appendix 1 details how we compute the Lagrangian multipliers and the corresponding $\widetilde{\cos}$ for these points. A through E give $\widetilde{\cos} \approx 1$, but F gives $\widetilde{\cos} \approx 0.52$ because $-\nabla_0$ and ∇_1 point into very different directions so F does not satisfy the KKT conditions.



Figure 1: Example with inputs x_1 and x_2 , goal output w_0 , constrained outputs w_1 and w_2 , dotted feasible area, and special points A through F

If no constraint is binding at the optimum point \boldsymbol{x}_* , then (10) should hold. To quantify how well this condition holds, we cannot use $\widetilde{\cos}$ (defined in (11)) with $\widetilde{\nabla}_0(\boldsymbol{x}_*)$ replaced by **0**; this replacement would give a denominator with the factor $||\widetilde{\nabla}_0(\boldsymbol{x}_*)|| = 0$. Obviously, (10) holds exactly at \boldsymbol{x}_* if $\partial w_0 / \partial x_j(\boldsymbol{x}_*) = 0$, $\forall j$. Therefore, we replace (11) by

$$d_{0; EGO}(\boldsymbol{x}_{*}) = \frac{1}{\max_{j}[|\partial w_{0}/\partial x_{j}(\boldsymbol{x}_{*})|]},$$
(12)

where the symbol $d_{0; EGO}$ is a mnemonic for (partial) derivative.

Unfortunately, black-box simulation implies that the $w_h(\boldsymbol{x})$ are unknown. Simulation does give the I/O data $(\boldsymbol{x}_i, w_{i;h})$, which we can use to estimate *metamodels* that approximate $w_h(\boldsymbol{x})$ and $\nabla_h(\boldsymbol{x})$. Actually, we use Kriging metamodels— as we explain in the next section.

4 Kriging in simulation

Like most publications on Kriging, we assume that k (number of inputs) is small, (say) $1 \le k \le 20$. Otherwise, we must face the well-known *curse of dimensionality*. Actually, our numerical examples have k = 2, 3, or 6. Subsection 4.1 discusses mean functions $E[y(\boldsymbol{x})]$ and covariance functions or kernels $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$, and estimators $\hat{y}(\boldsymbol{x})$ of $w(\boldsymbol{x})$ and their estimated standard error $s[\hat{y}(\boldsymbol{x})]$. Subsection 4.2 derives the estimators $\nabla[\hat{y}(\boldsymbol{x})]$ —or briefly $\hat{\nabla}(\boldsymbol{x})$ —of $\nabla(\boldsymbol{x})$. Subsection 4.3 summarizes *leave-one-out- cross-validation* (LOO-CV) for the selection of the "best" combination of $E[y(\boldsymbol{x})]$ and $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$.

4.1 Kriging: mean functions and kernels

We use univariate Kriging; i.e., we treat all t outputs as being independent. Univariate Kriging drastically simplifies the analysis, and may be superior because multivariate Kriging—or co-Kriging—requires the specification of a positive-definite covariance matrix and the estimation of extra covariances; namely, the individual cross-covariances between the t outputs in that matrix; see [24]. Theoretical and empirical comparisons of univariate and multivariate Kriging are also presented in [25], showing that multivariate Kriging may indeed be inferior. Univariate Kriging is used in nearly all publications that we reviewed in Section 2. Because we use univariate Kriging, we drop the subscript h for \hat{y}_h in this section—if h is not strictly necessary.

For each output, univariate Kriging requires the selection of $E[y(\boldsymbol{x})]$ and $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$ where \boldsymbol{x}' is a point in the k-dimensional input space that may coincide with \boldsymbol{x} . Obviously, if $\boldsymbol{x} = \boldsymbol{x}'$, then $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$ $= Var[y(\boldsymbol{x})] = \tau^2$ where τ^2 is the usual symbol for $Var[y(\boldsymbol{x})]$ in Kriging. This $E[y(\boldsymbol{x})]$ and $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$ have (hyper) parameters that we collect in the vector (say) $\boldsymbol{\psi}$. Popular specifications of $E[y(\boldsymbol{x})]$ are: (i) A constant which is also known as a zero-order polynomial β_0 ; this specification is called ordinary Kriging (OK). (ii) A first-order polynomial in \boldsymbol{x} (say) $\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}$ with $\boldsymbol{\beta}' = (\beta_1, \dots, \beta_k)'$, which is a type of universal Kriging (UK). (iii) A second-order polynomial, which is another type of UK. Obviously, OK requires the estimation of fewer parameters than UK does. The additional parameters of UK often lead to overfitting, so $s[(\hat{y}(\boldsymbol{x})]$ is higher for UK than it is for OK; see the references in [26], p. 198. We focus on OK, but in Appendix 2 we also investigate UK; we shall discuss this appendix, in Sections 4.2 and 4.3.

Well-known specifications of $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$ include the *Gaussian* (or squared exponential) kernel and the *Matérn* class. We give the well-known formulas for these kernels, because some Kriging software does not give $\widehat{\nabla}$ for this *Matérn* class so we derive the formulas for $\widehat{\nabla}$ ourselves. The most popular kernel in simulation (as opposed to geostatistics and machine learning) is the *anisotropic Gaussian* kernel, which we abbreviate to *G-kernel*. This kernel has the parameter vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_k)'$ with $\theta_j \ge 0$:

$$\rho(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta}) = \prod_{j=1}^{k} \exp \left[-\theta_j (x_j - x'_j)^2\right].$$
(13)

Obviously, this kernel is *separable*, and ψ becomes the (2 + k)-dimensional vector $(\beta_0, \tau^2, \theta_1, ..., \theta_k)'$. Using [27], p. 85, 89, 106, we derive the *anisotropic Matérn* kernel for v = 3/2—which we abbreviate to M-3/2—and v = 5/2—abbreviated to M-5/2:

$$r = r(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta}) = \sqrt{\frac{\sum_{j=1}^{k} (x_j - x'_j)^2}{\theta_j^2}}$$

$$\rho(v = 3/2, r) = \left(1 + \sqrt{3}r\right) exp\left(-\sqrt{3}r\right)$$

$$\rho(v = 5/2, r) = \left(1 + \sqrt{5}r + \frac{5r^2}{3}\right) exp\left(-\sqrt{5}r\right).$$
(14)

These kernels are *non-separable*, which affects the estimation of ψ ; see Section 5. We shall derive $\widehat{\nabla}$ for (13) and (14), in Section 4.2.

It is hard to select a specific combination of $E[y(\boldsymbol{x})]$ and $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$, because practical simulation models imply *implicit* specifications of $w_h(\boldsymbol{x})$. Consequently, we do not know the characteristics of these $w_h(\boldsymbol{x})$. We do assume that these $w_h(\boldsymbol{x})$ are differentiable at least once (so ∇_h , $\forall h$ exist), in the search domain $f_g(\boldsymbol{x}) \leq c_g$, $\forall g$ (see again Section 3). Therefore, we should select combinations of $E[y_h(\boldsymbol{x})]$ and $Cov[y_h(\boldsymbol{x}, \boldsymbol{x}')]$ such that the resulting $\hat{y}_h(\boldsymbol{x})$ is also differentiable at least once. Appendix 2 assumes that we do know the *degree of differentiability* of $w_h(\boldsymbol{x})$; so, we should select Kriging models that have the same characteristic degree ([11], p. 602 speaks of "ensuring the conformity", in the related but different context of utility functions in EGO). [27], p. 85 states that—for the Matérn class— $y(\boldsymbol{x})$ is (say) k_v times mean-square differentiable if and only if $v > k_v$. Hence, $y(\boldsymbol{x})$ is non-differentiable for a Matérn kernel with v = 1/2. The selection of a good combination of $E[y_h(\boldsymbol{x})]$ and $Cov[y_h(\boldsymbol{x}, \boldsymbol{x}')]$ is also discussed in Chapter 5 of [27]. Furthermore, [4], p. 44 mentions the ambiguity in the selection of a probabilistic model of the objective function (and the constrained functions $w_{h'}$ we would add).

To estimate ψ in (13) and (14), most authors use maximum likelihood estimation (MLE), which gives ψ . The computation of $\hat{\psi}$ is challenging, so most authors use standard Kriging software for this computation. Actually, we use a free-of-charge MATLAB toolbox called *design and analysis of computer experiments* (DACE), which is well documented in [28] and can be obtained from http://www2.imm.dtu.dk/pubdb/pubs/1460-full.html. DACE allows its users to choose among three mean functions and seven kernels; namely, the three $E[y(\boldsymbol{x})]$ specifications that we presented above and seven anisotropic kernels including the Gaussian—but excluding the *Matérn* class. Matérn kernels are available in the *MathWorks* software on https://ch.mathworks.com/help/stats/regressiongp.html. All Kriging software gives $\hat{y}(\boldsymbol{x})$ and $s[\hat{y}(\boldsymbol{x})]$, after the users provide (\boldsymbol{x}_i, w_i) . DACE also gives $\hat{\nabla}(\boldsymbol{x})$ (see [28], Eq. (2.18)). Therefore, we now derive $\hat{\nabla}(\boldsymbol{x})$ for the G-kernel and for the M-3/2 and M-5/2 kernels.

4.2 Kriging estimators of output gradients

In Appendix 2 we detail the derivation of $\widehat{\nabla}(\boldsymbol{x})$ for nine combinations of $E[y(\boldsymbol{x})]$ and $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$; namely, OK and two types of UK, combined with the G-kernel, the M-3/2 kernel, and the M-5/2 kernel. However, in this subsection we focus on the most popular specification; namely, OK with G-kernel. $(\widehat{\nabla}(\boldsymbol{x})$ is also discussed in [29], albeit not in an EGO context.)

We define the $n \times n$ covariance matrix $\mathbf{Cov}[y(\boldsymbol{x}_i, \boldsymbol{x}_{i'})] = \widehat{\boldsymbol{\Sigma}}$ with i, i' = 1, ..., n where n denotes the number of old simulated I/O combinations (\boldsymbol{x}_i, w_i) . Furthermore, we define the related n-dimensional covariance vector $\boldsymbol{\sigma}(\boldsymbol{x}_*) = Cov[y(\boldsymbol{x}, \boldsymbol{x}_*)]$. Finally, we define $\mathbf{w} = (w_1, ..., w_n)'$ and $\mathbf{1}_n = (1, ..., 1)'$. OK plugs $\widehat{\boldsymbol{\psi}}$ (which determines $\widehat{\beta}_0, \, \widehat{\boldsymbol{\sigma}}(\boldsymbol{x}_*)$ and $\widehat{\boldsymbol{\Sigma}}$) into the formula that assumes a known $\boldsymbol{\psi}$, which gives

$$\widehat{y}(\boldsymbol{x}_*) = \widehat{\beta}_0 + \widehat{\boldsymbol{\sigma}}(\boldsymbol{x}_*)' \widehat{\boldsymbol{\Sigma}}^{-1}(\mathbf{w} - \widehat{\beta}_0 \boldsymbol{1}_n).$$
(15)

We note that plugging-in $\hat{\psi}$ implies that $\hat{y}(\boldsymbol{x}_*)$ —defined in (15)—is a *nonlinear* estimator. Consequently, the true *mean squared prediction error* (MSPE) is underestimated by

$$s^{2}[\widehat{y}(\boldsymbol{x}_{*})] = \widehat{\tau}^{2} - \widehat{\sigma}(\boldsymbol{x}_{*})'\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\sigma}(\boldsymbol{x}_{*}) + \frac{[1 - \mathbf{1}_{n}'\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\sigma}(\boldsymbol{x}_{*})]^{2}}{\mathbf{1}_{n}'\widehat{\boldsymbol{\Sigma}}^{-1}\mathbf{1}_{n}}.$$
(16)

To estimate the true MSPE, we might use *parametric bootstrapping* and *conditional simulation*; see the references in [26], p. 209. However, we stick to (15) and (16)—computed via Kriging software—as most authors on EGO do.

In (15), all factors depend only on the old simulation data, except for $\hat{\sigma}(x_*)$; so, we can write

$$\nabla[\widehat{y}(\mathbf{x}_*)] = \nabla[\widehat{\mathbf{c}}'\widehat{\boldsymbol{\sigma}}(\mathbf{x}_*)] = \nabla[\widehat{\mathbf{c}}'\widehat{\tau}^2\widehat{\boldsymbol{\rho}}(\mathbf{x}_*,\mathbf{x})] \text{ with } \widehat{\mathbf{c}} = \widehat{\boldsymbol{\Sigma}}^{-1}(\mathbf{w} - \widehat{\beta}_0 \mathbf{1}_n)$$
(17)

For the G-kernel we obtain from (13) (suppressing the dependence of $\hat{\rho}$ on $\hat{\theta}$, to simplify the notation):

$$\frac{\partial\widehat{\rho}(\mathbf{x}_{*},\mathbf{x})}{\partial x_{*;j}} = -2\widehat{\theta}_{j}(x_{*;j} - x_{i;j})exp\left[\sum_{j'=1}^{k} -\widehat{\theta}_{j'}(x_{*;j'} - x_{i;j'})^{2}\right].$$
(18)

For the M-3/2 kernel and the M-5/2 kernel, we use the chain rule $(\partial \hat{\rho} / \partial \hat{r}) \times (\partial \hat{r} / \partial \hat{x}_{*,j})$ to obtain

$$\frac{\partial \hat{\rho}(v=3/2,\hat{r})}{\partial x_{*;j}} = \frac{-3\hat{r}exp\left(-\sqrt{3}\hat{r}\right)\left(\frac{x_{*;j}-x_{i;j}}{\hat{\theta}_{j}^{2}}\right)}{\sqrt{\frac{\sum\limits_{j'=1}^{k}(x_{*;j'}-x_{i;j'})^{2}}{\hat{\theta}_{j'}^{2}}}}$$

$$\frac{\partial \hat{\rho}(v=5/2,\hat{r})}{\partial x_{*;j}} = \frac{-\frac{5}{3}\hat{r}\left(1+\sqrt{5}\hat{r}\right)exp\left(-\sqrt{5}\hat{r}\right)\left(\frac{x_{*;j}-x_{i;j}}{\hat{\theta}_{j}^{2}}\right)}{\sqrt{\frac{\sum\limits_{j'=1}^{k}(x_{*;j'}-x_{i;j'})^{2}}{\hat{\theta}_{j'}^{2}}}}.$$
(19)

Obviously, (18) and (19) give the component (i, j) of the $n \times k$ matrix $\nabla \hat{\rho}(\mathbf{x}_*, \mathbf{x})$. Combing with (17) gives

$$\nabla[\hat{y}(\mathbf{x}_*)] = \hat{\tau}^2 \nabla \hat{\rho}(\mathbf{x}_*, \mathbf{x})' \hat{\mathbf{c}}.$$
(20)

4.3 Cross-validation for selection of mean functions and kernels

In Appendix 2 we compare (i) the combination of OK with a G-kernel per output type h, and (ii) the combination that is *automatically* selected from the nine combinations of $E[y(\boldsymbol{x})]$ and $Cov[y(\boldsymbol{x}, \boldsymbol{x}')]$. This comparison is made after the initial simulation I/O data $(\boldsymbol{x}_i, w_{i;h})$ (with $i = 1, ..., n_0$) have become available. For this automatic selection we adapt [30]'s LOO-CV; i.e., per output type h, we estimate which combination minimizes the median of $|\text{PES}_{i;h}| = |w_{i;h} - \hat{y}_{-i;h}| / s(\hat{y}_{-i;h})$ where $\hat{y}_{-i;h}$ denotes the Kriging predictor—for $w_{i;h}$ —computed after dropping $(\boldsymbol{x}_i, w_{i;h})$. Obviously, LOO-CV does not require new—possibly computationally expensive—simulation data, besides the old initial data. For fast computation of CV, we refer to [31].

This appendix shows that for the toy example (with t = 3) the best combination is OK with M-3/2, in all five macroreplications—except for macroreplication 1 where OK with G-kernel and OK with M-5/2 kernel give the same best combination for w_1 and w_2 as OK with M-3/2 kernel does. However, after the initial design, the fixed combination of OK with G-kernel gives lower sample medians of w_0 after (say) ten iterations and non-overlapping *interquartile ranges* (IQRs) after (say) forty iterations. The constrained Hartmann-6 example has t = 2 output functions $w_h(x)$ that are infinitely differentiable. The initial design does not give a clear pattern. Our fixed combination of OK and G-kernel for both outputs gives better w_0 -values for the first (say) 15 iterations; the combinations (i) and (ii) give the same w_0 -value after (say) 110 iterations. Altogether, we see no strong reasons to replace our combination of OK with G-kernels; most authors on EGO also use this combination, but other authors may use their own preferred combination.

We could have applied our LOO-CV not only after the initial design, but after each iteration. However, we leave that generalization for future research. We also refer to [32] and [33].

5 Initial design

Though EGO methods are sequential, they require an *initial design* that determines the input combinations that are needed to compute the initial values of $\hat{\psi}_h$, $\hat{y}_h(\boldsymbol{x}_*)$, $s^2[\hat{y}_h(\boldsymbol{x}_*)]$, and $\hat{\nabla}_h(\boldsymbol{x}_*)$. Subsection 5.1 discusses the type of initial design, focusing on *Latin hypercube sampling* (LHS). Subsection 5.2 discusses the *number* of initial points n_0 . Subsection 5.3 discusses initial designs that turn out to give no feasible points.

5.1 Type of initial design

We select the most popular design type in Kriging—namely, LHS. More specifically, we use *LHS with* midpoints, which fixes the distances among the n_0 values projected onto the k axes of \boldsymbol{x}_i ; i.e., per axis, the midpoints lie at distances that are multiples of $1/n_0$. So, these midpoints are not clustered. We expect that these midpoints give better $\hat{\psi}_h$ in the G-kernel (defined in (13)), which is separable. We conjecture that these midpoints also give better $\hat{\psi}_h$ in the M-3/2 kernel and the M-5/2 kernel. Appendix 2 demonstrates that these Matérn kernels can give numerical problems after the initial design (because new points may cluster); i.e., the covariance matrixes $\hat{\Sigma}_h$ —used to compute $\hat{y}_h(\boldsymbol{x}_*)$, $s[\hat{y}_h(\boldsymbol{x}_*)]$, and $\hat{\nabla}(\boldsymbol{x}_*)$ —are *ill-conditioned*. LHS with midpoints is an option in MATLAB's function *lhsdesign*.

Besides LHS, there are other *space-filling designs*, which are applicable in global modeling such as Kriging. For example, [34] discusses many types of space-filling designs, including 197 references. [29], [35], and [36] also discuss space-filling designs. [37] discusses composite grid designs for Kriging models.

5.2 Number of initial points

LHS does not imply a mathematical relationship between n_0 and k. Yet—given k—a higher n_0 results in a lower $s^2[\hat{y}_h(\boldsymbol{x}_*)]$. Actually, we select

$$n_0 = \frac{(k+1)(k+2)}{2} \text{ if } k \le 6; \text{ else } n_0 = 5k,$$
(21)

following [38]'s rule for selecting n_0 in sequential optimization in simulation via Kriging without EGO; for n = 5k, [38] refers to [18].

The literature presents several other rules for selecting n_0 . For example, [39] proposes $n_0 = 10k$, as a rule-of-thumb for "one-shot" (non-sequential) designs in sensitivity analysis (not optimization) via Kriging; that rule is also used in [40] for sequential optimization in random simulation. Furthermore, [41] states that 5k to 10k is often used, and also refers to [42]. For sequential optimization in simulation with k = 2, [10] uses $n_0 = 20$ in the "modified Townsend" example but uses $n_0 = 10$ in the toy example.

5.3 Initial designs without feasible points

Most publications—including [1], [14], and [10]—implicitly assume that at least one initial point is *feasible*. All initial points do satisfy all *input* constraints (so, $f_g(\mathbf{x}) \leq c_g, \forall g$). However, some problems may turn out to have not a single initial point that satisfies all output constraints; e.g., the spring example in Section 7.2 has a *very small feasible* area that—using the white-box equations—we estimate to be 9.7% of the total experimental area; so, an initial design with $n_0 = 10$ (see (21) with k = 3) has an expected number of feasible points less than one, and is quite likely to have no feasible point at all.

In practice, we may ask the users to specify a point that is feasible (albeit with a high value for w_0); e.g., the current system—if it already exists—provides a feasible suboptimal point. However, in our numerical experiments we do not need such human intervention; i.e., we use the following simple method to estimate a feasible point that is compatible with the initial simulation I/O data ($x_i, w_{i;h}$). (In Section 7 we shall see that we also use this method for the competing algorithms—defined in [14] and [10]—so we help these competitors to get started with their search for the optimum.)

Suppose that the initial design with n_0 —defined by (21)—points gives no feasible points. Then our algorithm and the competing algorithms obtain a larger LHS design with 10k points. We use LHS without midpoints (so the inputs are continuous); LHS with midpoints may imply that a small feasible area does not overlap any of these midpoints. In our examples, the experimental areas are defined by box constraints (instead of the more general $f_g(\mathbf{x}) \leq c_g, \forall g$ in (8), which may complicate the selection of an initial design). The original initial design with n_0 points enables the algorithm to compute $\hat{\psi}_h$. So, the algorithms can compute $\hat{y}_h(\mathbf{x}_i)$ with i = 1, ..., 10k. Finally, the algorithms determine the best feasible point among these 10k points:

$$\widehat{y}_{0;\min} = \min_{1 \le i \le 10k} [\widehat{y}_0(\boldsymbol{x}_i): \, \widehat{y}_{h'}(\boldsymbol{x}_i) \le c_{h'}, \, \forall h'],$$
(22)

if at least one of these 10k points is estimated to be feasible; else the algorithms obtain another LHS design with 10k points. If the experimental area has a non-empty feasible subarea, then this method does find $\hat{y}_{0; \text{ min}}$. So, the algorithms can return $w_{0; \text{ min}} = \hat{y}_{0; \text{ min}}$; see (3).

6 EGO and KKT conditions

We present our novel algorithm in Algorithm 1. Some steps of this algorithm are discussed in the following subsections. Subsection 6.1 estimates which output constraints are binding at \boldsymbol{x} . Subsection 6.2 derives our novel acquisition function $a_{\text{KKT}}(\boldsymbol{x})$. Subsection 6.3 presents several methods for finding the point that maximizes $a_{\text{KKT}}(\boldsymbol{x})$. Subsection 6.4 summarizes some other steps of the algorithm.

Algorithm 1 Pseudocode for the KKT algorithm

- 1: Select algorithm's control variables $\alpha_{BC; 0}$, $\alpha_{BC; \min}$, s_{\max}
- 2: Select n_0 , and use LHS with midpoints to sample an $n_0 \times k$ initial design matrix **X** with n_0 simulation inputs $\mathbf{x}_i = (x_{i;1}, ..., x_{i;k})$ such that $f_g(\mathbf{x}_i) \leq c_g, \forall g$, and $i = 1, ..., n_0$
- 3: Set $n \leftarrow n_0$
- 4: Set iteration number $s \leftarrow 0$
- 5: Obtain the $n \times t$ matrix with simulation outputs **W** by simulating $\mathbf{x}_i, \forall i$
- 6: while $s \leq s_{\max} \, \mathbf{do}$
- 7: Compute ψ
- 8: Find $\mathbf{x}_{\min} = \underset{1 \le i \le n}{\operatorname{argmin}} \{w_0(\mathbf{x}_i) : w_{h'}(\mathbf{x}_i) \le c_{h'}, \forall h', f_g(\mathbf{x}_i) \le c_g, \forall g\}$, and $w_{0;\min} = w_0(\mathbf{x}_{\min})$; if there is
 - no feasible \mathbf{x}_i , use LHS without midpoints to determine $w_{0;\min}$ (see Algorithm 2)
- 9: Set $\alpha_{BC} \leftarrow \alpha_{BC;0}$
- 10: Find $\widehat{\mathbf{x}}_{o}$ (see Algorithm 3)
- 11: Add $\widehat{\mathbf{x}}_{o}$ to \mathbf{X} , and update $n \leftarrow n+1$
- 12: Obtain $w_h(\widehat{\mathbf{x}}_o), \forall h$, by simulating $\widehat{\mathbf{x}}_o$, and augment W
- 13: Present the current best input \mathbf{x}_{\min} and its goal output $w_{0;\min}$
- 14: Update $s \leftarrow s + 1$
- 15: end while

Algorithm 2 Finding $w_{0:\min}$ through LHS in case of infeasible inputs

1: Input: $\widehat{\psi}$ 2: Set $w_{0;\min} \leftarrow \infty$ 3: while $(w_{0;\min} == \infty)$ do 4: Use LHS without midpoints to sample 10k points, satisfying $f_g(\mathbf{x}_i) \leq c_g, \forall g$ 5: Find $\widehat{y}_{0;\min} = \min_{1 \leq i \leq 10k} {\widehat{y}_0(\mathbf{x}_i) : \widehat{y}_{h'}(\mathbf{x}_i) \leq c_{h'}, \forall h'}$ 6: if $(\widehat{y}_{0;\min} \neq \infty)$ then 7: Set $w_{0;\min} = \widehat{y}_{0;\min}$ 8: end if 9: end while 10: Output: $w_{0:\min}$

Algorithm 3 Finding $\hat{\mathbf{x}}_{o}$

1: Inputs: $\widehat{\psi}$, $w_{0;\min}$, α_{BC} , $\alpha_{BC;\min}$ 2: Set $\widehat{\mathbf{x}}_{o} \leftarrow \emptyset$ and $A_{\lambda}(\mathbf{x}) \leftarrow \emptyset$ while $(\widehat{\mathbf{x}}_{o} == \emptyset \text{ and } \alpha_{BC} \geq \alpha_{BC; \min})$ do 3: for h' = 1 : t - 1 do 4: $\begin{array}{l} \text{if} \left(\frac{|\widehat{y}_{h'}(\mathbf{x}) - c_{h'}|}{s[\widehat{y}_{h'}(\mathbf{x})]} \leq z_{1-\alpha_{\text{BC}}/[2(t-1)]} \right) \text{ then} \\ \text{Update } A_{\lambda}(\mathbf{x}) \leftarrow A_{\lambda}(\mathbf{x}) \cup \{h'\} \end{array}$ 5:6: end if 7:end for 8: if $A_{\lambda}(\mathbf{x}) \neq \emptyset$ then 9: 10: Find $A_{\mu}(\mathbf{x})$ at current \mathbf{x} Compute $\widehat{\nabla}_{h''}(\mathbf{x})$ for $h'' \in A_{\lambda}(\mathbf{x})$ and $\widehat{\nabla}_0(\mathbf{x})$ at current \mathbf{x} 11:Solve $\widehat{\mathbf{x}}_{o} = \operatorname{argmax} \left\{ \widehat{\operatorname{EI}}_{0}(\mathbf{x}) \times \widehat{\operatorname{cos}}(\mathbf{x}) : \widehat{y}_{h'}(\mathbf{x}) + z_{1-(\alpha_{\mathrm{BC}}/(t-1))}s[\widehat{y}_{h'}(\mathbf{x})] \le c_{h'}, \forall h', f_{g}(\mathbf{x}) \le c_{g}, \forall g \right\}$ 12:else 13:Set $\alpha_{\rm BC} \leftarrow \alpha_{\rm BC}/2$ 14: end if 15:16: end while 17: Output: $\widehat{\mathbf{x}}_{o}$

6.1 Estimation of binding output constraints

The KKT conditions in (9) use $A_{\lambda}(\boldsymbol{x})$ (the set with the indexes h'' at \boldsymbol{x}), so our algorithm needs to estimate which output constraints are binding. Let $\hat{\boldsymbol{x}}_{o}$ denote the estimated optimal (subscript o) point that is to be simulated in the next iteration. We use *confidence intervals* (CIs) around $c_{h'}$ (threshold for $w_{h'}$) that are *symmetric* and *two-sided* (see |.| and the factor 2 in (23) below), because we wish $\hat{\boldsymbol{x}}_{o}$ to lie exactly on the boundary. Furthermore, we use a *familywise* error rate α_{BC} (BC stands for binding constants); i.e., we apply *Bonferroni*'s inequality, so we use a *per-comparison* rate $\alpha_{BC}/(t-1)$ per output constraint. We test *all* (t-1) estimated constrained outputs $\hat{y}_{h'}(\hat{\boldsymbol{x}}_{o})$, but we include h' in $A_{\lambda}(\hat{\boldsymbol{x}}_{o})$ only if the CI for $\hat{y}_{h'}(\hat{\boldsymbol{x}}_{o})$ covers $c_{h'}$. Obviously, $s[\hat{y}_{h'}(\hat{\boldsymbol{x}}_{o})]$ denotes the square root of $s^2[\hat{y}_{h'}(\hat{\boldsymbol{x}}_{o})]$ that follows from (16). Altogether, we use the following decision rule:

If
$$\frac{|\widehat{y}_{h'}(\widehat{x}_{o}) - c_{h'}|}{s[\widehat{y}_{h'}(\widehat{x}_{o})]} \leq z_{1-\alpha_{BC}/[2(t-1)]}, \text{ then } h' \in A_{\lambda}(\widehat{x}_{o}).$$
 (23)

Typically, familywise error rates are higher than per-comparison rates are; also see [43]. In our examples we start with $\alpha_{BC} = 0.20$. If this α_{BC} -value leaves $A_{\lambda}(\hat{x}_{o})$ empty, then we decrease this value. Actually, in our algorithm we halve α_{BC} . If the new α_{BC} value still leaves $A_{\lambda}(\hat{x}_{o})$ empty, then we again halve this value—until the latest α_{BC} -value reaches a prespecified minimum value (say) $\alpha_{BC; \min}$. If $\alpha_{BC} = \alpha_{BC; \min}$ still gives an empty $A_{\lambda}(\hat{x}_{o})$, then we may infer that the problem has no binding constraints. We then switch from the KKT conditions to the condition that requires a zero goal-gradient; see (10).

In practice, simulation optimization may have no constrained outputs $w_{h'}$ at all; see the numerous publications on such problems. Nevertheless, many simulation optimization problems do have binding constraints; see the spring example in Section 7.2 and the truss example in Section 7.3. Our own research on logistic problems also reveals binding constraints; i.e., we try to minimize the costs of the logistic system, while meeting the threshold for the service provided by that system.

6.2 A novel acquisition function

Whereas our algorithm uses the *two-sided* CIs defined in (23) to estimate *which* output constraints are binding, our algorithm subsequently finds \hat{x}_{o} that is estimated to lie *inside* the feasible area (i.e., on the safe side of the estimated boundary) and *close* to that boundary. So, \hat{x}_{o} satisfies the following *one-sided* CIs:

$$\widehat{y}_{h'}(\widehat{x}_{o}) + z_{(1-\alpha_{BC}/(t-1))} \times s[\widehat{y}_{h'}(\widehat{x}_{o})] \le c_{h'}, \forall h'.$$

$$(24)$$

It may happen that $s[\hat{y}_{h'}(\hat{x}_{o})]$ is so high that—even if α_{BC} is relatively small—no \hat{x}_{o} within the experimental area satisfies (24). In that case, our algorithm replaces the safety factor $z_{(1-\alpha_{BC}/(t-1))}$ by 0 in (24).²

If the binding constraints and the gradients ∇_h were known, then (9) should hold. Actually, our algorithm uses (23) to estimate which constraints are binding, and uses $\widehat{\nabla}_h$ (which include $\widehat{\nabla}_0$ and $\widehat{\nabla}_{h''}$) to estimate ∇_h . Next the algorithm uses $\widehat{\nabla}_{h''}$, to compute the LS estimate $-\widetilde{\nabla}_0$. To estimate how well the KKT conditions hold, the algorithm computes $\cos(-\widehat{\nabla}_0, -\widetilde{\widehat{\nabla}}_0)$ via (11). We abbreviate $\cos(-\widehat{\nabla}_0, -\widetilde{\widehat{\nabla}}_0)$ to $\widetilde{\cos}$. Our algorithm uses this $\widetilde{\cos}$ as a multiplicative correction factor—or penalty—for \widehat{EI}_0 (this \widehat{EI}_0 was defined in Section 2.2.1). Altogether, our novel *acquisition function* is

$$a_{\rm KKT}(\boldsymbol{x}) = \widehat{\rm EI}_0(\boldsymbol{x}) \times \widetilde{\cos}(\boldsymbol{x})$$
⁽²⁵⁾

where our algorithm searches for x that satisfies the (t-1) constraints for the estimated outputs $\hat{y}_{h'}$ in (24) and the v input constraints $f_g(x) \leq c_g$.³

In summary, our algorithm searches within the estimated feasible area, which satisfies the one-sided CIs defined in (24) and the known input constraints $f_g(\boldsymbol{x}) \leq c_g$. Within this area, our algorithm selects the point that maximizes a_{KKT} defined in (25). In (25), $\widehat{\text{EI}}_0$ makes our algorithm prefer points with a high $\widehat{\text{EI}}_0$ -value; these points lie near the estimated boundary of the feasible area, because we assume that the global optimum lies on this boundary. The factor $\widehat{\cos}(\boldsymbol{x})$ makes our algorithm select the point near this boundary that is estimated to best satisfy the KKT conditions. To compute $\widehat{\cos}$, our algorithm must estimate which constraints are binding; therefore, our algorithm uses the two-sided CIs defined in (23).

Before we further detail our algorithm, we briefly compare our $a_{\text{KKT}}(\boldsymbol{x})$ with the acquisition functions of the two competing algorithms.

(i) [14] replaces $\widehat{\text{EI}}_0$ by $\widehat{\text{PI}}_0$, and multiples this $\widehat{\text{PI}}_0$ by $\widehat{\text{PF}}$. Our algorithm uses the more popular $\widehat{\text{EI}}_0$ ([10] also uses $\widehat{\text{EI}}_0$; see (7)). Both $\widehat{\text{PF}}$ and $\widehat{\cos}(\boldsymbol{x})$ have values within the interval [0, 1]. However, $\widehat{\text{PF}}$ includes all output constraints (binding or non-binding) and $\widehat{\text{PF}}$ decreases as the number of constraints (t-1) increases.

(ii) [10]'s $a_{BF}(\boldsymbol{x})$ (defined in (7)) adds an extra term to \widehat{EI}_0 ; this term is a penalty for approaching the estimated boundary. So, both [10]'s algorithm and our algorithm search within the estimated feasible area. We note that [10]'s penalty term includes *all* output constraints, like [14] does. Moreover, $a_{BF}(\boldsymbol{x})$ may mix different measurement scales used by $\widehat{y}_h(\boldsymbol{x})$, for different *h*. Finally, that algorithm solves an unconstrained

²Even if $\hat{\boldsymbol{x}}_{o}$ satisfies (24)—or satisfies (24) with a zero safety-factor—then the next simulation run with $\hat{\boldsymbol{x}}_{o}$ as input may turn out to be infeasible. Then the new simulation I/O data $(\hat{\boldsymbol{x}}_{o}, w_{h}(\hat{\boldsymbol{x}}_{o}))$ —added to the old data—improves the accuracy of $\hat{\boldsymbol{\psi}}$, but $w_{0}(\hat{\boldsymbol{x}}_{o})$ does not improve $w_{0; \min}$.

³The computation of $\widehat{\mathrm{El}}_{0}(\boldsymbol{x})$ in (2) gives numerical problems if $s[\widehat{y}_{0}(\boldsymbol{x})]$ is close to zero (which happens if \boldsymbol{x} is close to an old point \boldsymbol{x}_{i}). Actually, if $s[\widehat{y}_{0}(\boldsymbol{x})] < 10^{-5}$, then our algorithm uses $\phi(\boldsymbol{x}) = 0$ and either $\Phi(\boldsymbol{x}) = 0$ if $w_{0; \min} - \widehat{y}_{0}(\boldsymbol{x}) < 0$ (so $\widehat{\mathrm{El}}_{0}(\boldsymbol{x}) = 0$) or $\Phi(\boldsymbol{x}) = 1$ if $w_{0; \min} - \widehat{y}_{0}(\boldsymbol{x}) \geq 0$ (so $\widehat{\mathrm{El}}_{0}(\boldsymbol{x}) = w_{0; \min} - \widehat{y}_{0}(\boldsymbol{x})$).

problem, whereas our algorithm solves a constrained problem as it uses (24) and input constraints.

For problems with optimal solutions *inside* their feasible areas, we replace $\widehat{\cos}(\boldsymbol{x})$ by $\widehat{d}_{0; EGO}(\boldsymbol{x})$, which follows from (12) replacing $w_0/\partial x_j$ by $\widehat{y}_0/\partial x_j$:

$$a_{0; EGO}(\boldsymbol{x}) = \widehat{\mathrm{EI}}_{0}(\boldsymbol{x}) \times \widehat{d}_{0; EGO}(\boldsymbol{x}).$$
(26)

6.3 Maximization of the acquisition function

In general, EGO uses different methods to maximize their acquisition functions. The conceptually simplest method uses LHS to generate a large set of *candidate points*, computes \hat{y}_h at these points, and finds the point that maximizes a(x); e.g., [16], p. 308 uses 1000 candidates for the toy problem (with k = 2) and [16], p. 310 discusses the size of the LHS sample for the "welded beam" example with k = 4. Obviously, a set with candidates implies *discretization* of the experimental area.

To avoid such a discretization, we may apply numerical optimizers. These optimizers are either global or local. Global optimizers include evolutionary algorithms, which include genetic algorithms; references are given in [26], p. 268. Local optimizers may get trapped in local optima, because $\widehat{EI}_0(\mathbf{x})$ has many local optima; i.e., old points \mathbf{x}_i imply $s[\widehat{y}_0(\mathbf{x}_i)] = 0$ (i = 1, ..., n), so there are n local minima $\widehat{EI}_0(\mathbf{x}_i) = 0$ and \widehat{EI}_0 may be positive between these minima. Our algorithm uses MATLAB's pattern search (PS) with restarts. To sample these multiple starting points, our algorithm uses LHS without midpoints (because we wish to ensure that the probability of selecting an old point is zero; midpoints may give PS endpoints that are the same as the corresponding starting points). Our algorithm uses only 10 PS-restarts, to speed-up our numerical experiments. Our algorithm and [14]'s algorithm use the same 10 restarts.⁴ [10]'s algorithm also uses 10 restarts, but some restarts may require a slightly different method.⁵ We detail PS including our choice of PS options (e.g., mesh size), in Appendix 1.⁶

6.4 Some other steps

Like other EGO algorithms do, our algorithm uses $\hat{\boldsymbol{x}}_{o}$ to obtain $w_{h}(\hat{\boldsymbol{x}}_{o})$. The algorithm finds out whether $w_{h'}(\hat{\boldsymbol{x}}_{o})$ satisfies the threshold $c_{h'}$, $\forall h'$. If $\hat{\boldsymbol{x}}_{o}$ is feasible, then $w_{0}(\hat{\boldsymbol{x}}_{o})$ may (or may not) improve $w_{0; \min}$. Next, the algorithm uses all n available simulated I/O combinations, to re-estimate $\boldsymbol{\psi}$, etc.

⁴Initially, we used 10k starting points for an example with k = 6; namely, the Hartmann-6 example ([40] also uses 10k starting points selected by LHS, but uses fmincon instead of PS). Next, we find that 10 restarts give results similar to the results of 10k restarts.

⁵ [10]'s $a_{BF}(\boldsymbol{x})$ (defined in (7)) requires $\hat{y}_{h'}(\boldsymbol{x}) < 0$ so we define the problem such that $c_{h'} = 0$ (see again the text below (8)). Binding constraints imply that the feasible area is less than 100% of the experimental area. We denote the number of *acceptable* PS-starting points (with $\hat{y}_{h'}(\boldsymbol{x}) < 0$) by n_{acc} . We expect that LHS with 10 restarts gives $n_{acc} < 10$. Therefore, we use LHS without midpoint as defined below (22), and find the first $10 - n_{acc}$ acceptable points.

 $^{^{6}}$ PS gives better results than MATLAB's fmincon does, for the toy example. Therefore we decide to apply PS to all our examples.

Like other EGO algorithms, our algorithm does not test the *validity* of the estimated Kriging metamodels; i.e., the algorithm uses the fitted Kriging model only to guide the selection of the next point, after the initial design. As *n* increases, $s[\hat{y}_h(\boldsymbol{x}_*)]$ decreases. We conjecture that an inadequate Kriging model gives our algorithm a bad start, but the algorithm may still give a good final estimate of the global optima—albeit after a longer search than in case of a good start. This conjecture is not rejected by our numerical experiments.

Expensive simulation means that computing w_h for a given x requires so much time that impatient users must wait relatively long for the presentation of the final $w_{0; \min}$. A convergence plot for $w_{0; \min}$ (by definition) displays $w_{0; \min}(s)$ as a function of s = 0, 1, ..., where s = 0 corresponds with the initial design and s > 0 corresponds with iteration s. Users might inspect this plot while it is being built after each iteration, and stop the algorithm's search as soon as they are satisfied with the solution—at the risk of premature stopping. However, to eliminate the role of these users, we automate our experiments; i.e., we execute a few preliminary experiments with each example, and next we select a maximum number of iterations (say) s_{\max} per example such that each algorithm can converge to the true optimum goal value of that example. To estimate this true value, we apply PS to the white-box equations of that example.

We point out that [44], p. 787 states: "one may want to avoid spending too many evaluations on the infeasible regions, while exploring the regions close to the boundaries, where the optimum is likely to be". Similar advice is given in [19], [45], and [46]. Actually, our algorithm is compatible with that advice.

7 Numerical examples

We obtain numerical results for our algorithm and the two alternatives derived in [14] and [10], respectively. More specifically, we apply these three algorithms to five examples; namely, the toy example (in Section 7.1), the spring example (in Section 7.2), the truss example (in Section 7.3), and a constrained variant and an unconstrained variant of the Hartmann-6 example (in Section 7.4). For these five examples, we (not the three algorithms) know the white-box equations, so we can derive their true global minima.

As we mentioned in our literature review in Section 2, the essence of an EGO algorithm is its *acquisition* function $a(\mathbf{x})$. For a better comparison of the two alternative algorithms with our algorithm, we use their acquisition functions $a_{\text{PI-PF}}(\mathbf{x})$ and $a_{\text{BF}}(\mathbf{x})$ —but we use our algorithm for the selection of an initial LHS design with n_0 points, our univariate OK models with G-kernels, etc. So, for all three algorithms we use DACE to compute $\hat{\psi}^7$. If the initial design does not give a feasible solution, then we use our method for estimating a feasible solution (see Section 5.3). We apply PS—with 10 restarts—to estimate $\hat{\mathbf{x}}_o$ for the three

⁷To compute $\hat{\theta}_{h;j}$, we must select a search area—but we do not know a reasonable range for $\hat{\theta}_{h;j}$. After some trial-and-error, we select a small lower bound 0.001 and a large upper bound 10.

algorithms. Each algorithm uses the same 10 restarts—so, they use *common random numbers* (CRN)—for a specific iteration of the algorithm; however, different iterations use different restarts. The variable α_{BC} features only in our algorithm; we use the initial value $\alpha_{BC; 0} = 0.20$ in all five examples.

We stop all three algorithms when their number of iterations reaches s_{max} (see Section 6). Our numerical results show that all three algorithms have indeed converged at s_{max} , in all our examples; each example has its own s_{max} (also see the convergence plots in the subsections below). In expensive simulation, the EGO computations require negligible time; i.e., wall-clock time mainly depends on the total number of simulation observations that an EGO algorithm uses to search for the global optima.

We code all three algorithms in MATLAB; we are willing to share our code with the readers. To sample *pseudorandom numbers* (PRN), we use the generator called *mrg32k3a*, which can create 2^{63} independent substreams of length 2^{127} so we can control the seeds such that macroreplications use non-overlapping sub-streams; see [47] and https://www.mathworks.com/help/matlab/ref/randstream.randstream.create.html.

In Appendix 4 we present *individual* convergence plots per algorithm and per example. Each plot is the result of a macroreplication that starts with its own initial design, so that we can better compare algorithms (i.e., each macroreplication uses CRN for the initial LHS). In the following subsections we summarize these individual convergence plots, presenting the estimated medians and IQRs for $w_{0;\min}$ at selected iterations s. In each convergence plot, we also display the true global minimum. We select s_{\max} such that in each macroreplication, each algorithm has converged before s reaches s_{\max} ; i.e., the convergence plots end as horizontal lines. The number of macroreplications is 50 in all examples—except for the unconstrained Hartmann-6 example where we use only 10 macroreplications, which saves time and yet gives informative results (as we shall see).

7.1 Toy example

In Appendix 4 we present the *white-box* equations $w_h(\boldsymbol{x})$ and $l_j \leq x_j \leq u_j$ for the toy example, which gave Fig. 1. To estimate the relative size of the *feasible* area, we sample 100,000 points \boldsymbol{x} with $l_j \leq x_j \leq u_j$ —through LHS with midpoints—and use these points as inputs for $w_{h'}(\boldsymbol{x})$. This gives an estimated feasible area that is 46% of the total experimental area. This 46% gives a 98% probability of obtaining at least one feasible input and an expected number of feasible points equal to 2.76, for an initial LHS design with $n_0 = 6$. Actually, 46 of our 50 macroreplications give initial designs with at least one feasible point; for the other macroreplications, our algorithms obtain $\hat{y}_{0; \min}$ (defined in (22)) instead of $w_{0; \min}$. Appendix 4 shows selected iterations for macroreplication 1, which give more details.



Figure 2: Convergence plots: medians and quartiles for $w_{0;\min}$ at selected s, esimated from 50 macroreplications for three algorithms applied to the toy example



Figure 3: Final optimum input combination \mathbf{x}_{\min} for our algorithm (left pane), Carpio's algorithm [14] (middle pane), and Pourmohamad's algorithm [10] (right pane), in 50 macroreplications for the toy example

Fig. 2 displays convergence plots for the three algorithms. More precisely, these plots display the median goal value per iteration, and after every five new iterations these plots display the 25% and the 75% percentiles (or quartiles), which give the IQRs. For small iteration numbers, we estimate some medians and ranges from less than 50 macroreplications because we use $\hat{y}_{0; \text{ min}}$ instead of $w_{0; \text{ min}}$; e.g., in the initial design (or iteration 0), there are 4 out of 50 macroreplications without a feasible input. We infer: (i) In the early iterations, our algorithm converges slightly faster. (ii) All three algorithms converge to values close to $\boldsymbol{x}_{\text{A}}$. (iii) The IQRs of our algorithm are relatively big; this occurs because our algorithm converges to $\boldsymbol{x}_{\text{C}}$ in 5 out of 50 macroreplications, as Fig. 3 shows. Appendix 4 also displays 50 individual convergence plots and additional percentiles. Actually, we help both alternative algorithms in case they cannot find a feasible input (i.e., we let them use $\hat{y}_{0; \min}$); e.g., our convergence plot is better than Fig. 4 in [10].

Fig. 3 displays the final x_{\min} for the three algorithms, in 50 macroreplications. Our algorithm (left-most



Figure 4: Tension-compression spring example

pane) gives 5 macroreplications close to $\mathbf{x}_{\rm C}$ (as we mentioned above). In practice, the users might restart an algorithm using different initial designs and select the result of the best restart; actually, Fig. 3 shows 50 restarts, and its best restart gives $x_{\rm min}$ extremely close to the true optimum. In practice the users may decide to use either fewer than 50 restarts or a parallel computer (so the wall-clock time does not increase; it is simple to implement our algorithm on a parallel computer). In general, we recommend restarts for any local optimizer (for PS, we do use restarts, which do not require much computer time because PS uses \hat{y}_h instead of w_h).

Corresponding with Fig. 3, Appendix 4 displays a boxplot of the *slack* $c_1 - w_1(\boldsymbol{x})$ for our algorithm; the estimated median slack is 0.0013, so our algorithm gives final \boldsymbol{x}_{\min} -values that we expect to be feasible and close to the boundary.

7.2 Tension-compression spring example

Fig. 4 (based on [48]) displays the engineering view of the spring example, including the k = 3 inputs d, D, and N where d denotes the wire diameter, D the mean coil diameter, and N the number of "active" coils which implies that N is continuous. The input constraints are $0.05 \le d \le 0.20$, $0.25 \le D \le 1.30$, and $2 \le$ $N \le 15$. The goal output w_0 is the weight; the constraint w_1 is the minimum deflection of the spring caused by the axial loading, w_2 is the maximum shear stress in the wire that should be smaller than the allowable shear stress of the material, w_3 is the surge frequency that should be greater than a specified value, and w_4 is a limit for the outside diameter of the spring. Appendix 4 gives the white-box equations $w_h(\mathbf{x})$ with h= 0, ..., 4 and $x_1 = N$, $x_2 = D$, $x_3 = d$, and right-hand thresholds $c_{h'} = 0$; these equations are nonlinear



Figure 5: Convergence plots for spring example: medians and IQRs for $w_{0; \min}$ at selected iteration numbers s, estimated from at most 50 macroreplications for three algorithms

except for $w_4(\boldsymbol{x})$.⁸

Furthermore, [49] states that this example is a convex optimization problem. Hence, we do not expect that this problem benefits from EGO; i.e., a simpler algorithm would have sufficed. Nevertheless, this problem is popular in constrained global optimization; see the examples in [18]. Table 3 in [49] lists fifteen optimal points that are found in the literature. From that table we conclude that these points indeed give (virtually) the same goal value, but some points differ substantially. Next, we apply PS with 10,000 starting points to the white-box equations; this gives 154 different optimal points. The first point and the second point give two binding constraints (namely, the constraints 1 and 2). So, the decision rule in (23) may infer that several constraints are binding; the importance of multiple binding constraints is emphasized in [50]. Altogether, we conclude that the goal function is not strictly convex.

We estimate that the *feasible* area is nearly 10% of the total experimental area (for this estimate we use LHS with midpoints to sample 100,000 points \boldsymbol{x} that give $w_{h'}(\boldsymbol{x})$). Our initial design uses $n_0 = 10$. In our experiment, 11 of our 50 macroreplications give initial designs without any feasible point. For these 11 macroreplications the algorithms use $\hat{y}_{0;\text{min}}$ instead of $w_{0;\text{min}}$. We note that [14] and [10] do not apply their algorithms to this example, so they are not confronted with initial designs without any feasible solution.

After some preliminary experimentation, we fix s_{max} at 80; Fig. 5 shows that 80 is generous for our algorithm and [14]'s algorithm, whereas 80 hardly suffices for [10]'s algorithm. This graph displays estimated medians and IQRs, for each algorithm. The IQRs for our algorithm and for [14]'s algorithm are so short that they are invisible. These two algorithms perform very similarly. [10]'s algorithm performs relatively poorly;

⁸Different publications use different symbols, and include wrong equations; e.g., [48] misses the division sign (/) between the third factor and the fourth factor of its equation (26). [49] also mentions several problems.



Figure 6: Three-bar truss example

i.e. it has hardly converged after 80 iterations.

Because this example has more than two inputs, we cannot display the analogue of Fig. 3. However, Appendix 4 does display boxplots of the *slacks* for the binding constraints 1 and 2. These boxplots show that these slacks are always small and positive. Next we investigate an example with a bigger feasible area than the spring example.

7.3 Three-bar truss example

Fig. 6 displays the *engineering* view of the truss example with the cross-sectional areas $x_1 = A1 = A3$ and $x_2 = A2$, where w_0 is the volume of the (statically loaded) truss structure while accounting for three stress constraints $w_{h'}$ (with h' = 1, 2, 3) and the input constraints $0 \le x_1 \le 1$ and $0 \le x_2 \le 1$; see [18] and [51]. (Instead of three bars, [48] gives truss examples with ten, eighteen, and twenty-five bars.)

Appendix 4 gives the white-box nonlinear equations $w_h(\mathbf{x})$ (with h' = 0, 1, 2, 3) and the (known) input constraints that correspond with Fig. 6. Using these equations and input constraints, we derive Fig. 7 (which is the analogue of Fig. 1 for the toy example). Our graph shows a single global optimum \mathbf{x}_A and no local optima, so the feasible area is *convex* (like in the spring example). We apply PS to $w_h(\mathbf{x})$, and find $\mathbf{x}_A \approx (0.79, 0.41)'$. Next, we compute $w_h(\mathbf{x}_A)$, which includes $w_0(\mathbf{x}_A) \approx 263.90$ and shows that $w_1(\mathbf{x}_A)$ is binding. Furthermore, this graph shows that the iso-goal line $w_0(\mathbf{x}) \approx 263.90$ lies "close" to the boundary of the feasible area (defined by $w_1(\mathbf{x}_A) = 0$), so a change in \hat{x}_0 such that the resulting \hat{x}_0 lies "near" that boundary (on either the feasible or the infeasible side) gives a relatively small change in $w_0(\hat{x}_0)$ and hence in $\widehat{El}_0(\hat{x}_0)$. Appendix 4 shows selected iterations of macroreplication 1 with more details.

Fig. 8 displays convergence plots estimated from 50 macroreplications for three algorithms. These plots



Figure 7: Three-bar truss example with inputs x_1 and x_2 , goal output w_0 and constrained outputs w_1 , w_2 , and w_3 ; dotted area denotes feasible area



Figure 8: Convergence plots for truss example: medians and IQRs for $w_{0; \text{ min}}$ at selected iteration numbers s, estimated from 50 macroreplications for three algorithms



Figure 9: Final optimum input combination \mathbf{x}_{\min} for our algorithm (left pane), Carpio's algorithm [14] (middle pane), and Pourmohamad's algorithms [10] (right pane), in 50 macroreplications for the truss example

show that (i) our algorithm converges faster than the two alternative algorithms do and (ii) the IQRs of our algorithm and [10]'s algorithm become very short after iteration 10.

Fig. 9 displays \mathbf{x}_{\min} for our algorithm (left pane), [14]'s algorithm (middle pane), and [10]'s algorithm (right pane), in 50 macroreplications for the truss example; [10]'s \mathbf{x}_{\min} -values show the smallest variation, and [14]'s values show the highest variation. The boxplot of the *slack* for the binding constraint in Appendix 4 shows that the slack values are very small positive numbers.

7.4 Hartmann-6 examples

Finally, we investigate an example with both a global optimum and local optima (as in the toy example); after all, EGO was originally developed for such problems. The popular unconstrained Hartmann-6 example has k = 6 inputs (there are also variants with k = 3 or k = 4). The input constraints are $0 \le x_j \le 1$ with j = 1, ..., 6. This example has 52 constants ($\alpha_i, A_{i;j}$, and $P_{i;j}$) that define a nonlinear goal function $w_0(\boldsymbol{x}, \alpha_i, A_{i;j}, P_{i;j})$); see Appendix 4. To this example, [17] adds one output constraint:

$$w_1(\boldsymbol{x}) = \sqrt{\sum_{j=1}^{6} x_j^2} \le c_1.$$
(27)

Because our focus is on problems with *binding* output constraints, we replace [17]'s $c_1 = 1.25$ by our tighter threshold $c_1 = 0.946$. In Appendix 4, we use the white-box equations to derive the true global optimal point \boldsymbol{x}_A , which gives $w_0(\boldsymbol{x}_A) \approx -3.32$ and $w_1(\boldsymbol{x}_A) \approx 0.95$ (so, \boldsymbol{x}_A indeed lies on the boundary defined by (27) with $c_1 = 0.946 \approx 0.95$). Moreover, we derive one *local* optimum; namely, $w_0(\boldsymbol{x}_B) \approx -2.2719$ (> $w_0(\boldsymbol{x}_A) \approx$ -3.322), which also lies on this boundary. We estimate that the feasible area is less than 6% (we again use LHS with 100,000 points). Our initial LHS design has $n_0 = 28$ points (see (21)). So, the expected number of feasible initial points is less than 2; actually, 6 of the 50 macroreplications use $\hat{y}_{0;\min}$ instead of $w_{0;\min}$.



Figure 10: Convergence plots for constrained Hartmann-6 example: medians and IQRs for $w_{0; \text{ min}}$ at selected iteration numbers s, estimated from at most 50 macroreplications for three algorithms

To speed-up our experiment with this example (with k = 6 inputs, so its search area is relatively large), we stop each macroreplication after $s_{\text{max}} = 120$ iterations (after the initial design). We find that after these 120 iterations, α_{BC} (used by our algorithm) still has the initial value; namely, 20%.

Fig. 10 displays the medians and IQRs of the convergence plots for the three algorithms, estimated from at most 50 macroreplications; Appendix 4 displays the 50 individual plots and selected percentiles per algorithm. From these plots we infer: (i) Our algorithm converges to the local minimum seven times, whereas [14]'s and [10]'s algorithms converge to this local minimum one and five times, respectively. (ii) Our algorithm performs better than [10]'s, until iteration 75. (iii) [14]'s IQRs become very small, after iteration 50. (iv) The IQRs of our algorithm and [10]'s overlap, until iteration 115; i.e., these ranges do not differ significantly, until that iteration. Appendix 4 displays a boxplot of the *slack* $c_1 - w_1(\mathbf{x})$ with $c_1 = 0.946$, for our algorithm; the estimated median slack is 0.0096, so we expect that our algorithm gives final \mathbf{x}_{min} -values that are feasible and close to the boundary.

Finally, we obtain numerical results for the Hartmann-6 example with a non-binding output constraint; namely, (27) with $c_1 = 1.25$ (instead of our $c_1 = 0.946$; [17] communicated his correct c_1 in a personal e-mail). These results are detailed in Appendix 4, from which we infer: (i) Our algorithm with $a_{0; EGO}(x)$ (defined in (26)) converges faster to the global minimum than the two alternatives do. (ii) The performance of [10]'s algorithm is relatively poor. (iii) After iteration 50, our algorithm and [14]'s algorithm converge to the same estimated median goal value. (iv) After iteration 60, [14]'s algorithm gives IQRs that are almost zero.

8 Conclusions and future research

In this paper, we focussed on constrained optimization in deterministic simulation. We derived a novel algorithm that combines popular EGO with the KKT conditions, which account for constraints—for outputs and inputs—that are binding at the global optima. Our algorithm estimates which simulation input combination \boldsymbol{x} maximizes our novel acquisition function that multiplies the current Kriging (or GP) estimates of (i) EGO's EI(\boldsymbol{x}) and (ii) $\widetilde{\cos}(\boldsymbol{x})$ which quantifies how well the KKT conditions hold at \boldsymbol{x} , using a LS model for the KKT conditions. We obtained numerical results for several mathematical examples and mechanical-engineering examples, for our algorithm and two alternatives; these alternatives are recent representatives for either classic EGO algorithms or hybrid algorithms that combine EGO with MO. We used the white-box equations of the spring example and the truss example, and discovered that these examples have convex feasible areas. In the truss example, our algorithm converges faster to the true global optimum than the two alternative algorithms do; consequently, our algorithm can serve impatient customers in computation-ally expensive simulation. Altogether we conclude that our algorithm may be a suitable EGO variant for constrained optimization in deterministic simulation.

Our algorithm is flexible, because it is modular; i.e., it includes the following modules: (i) A specific statistical model; namely, OK combined with a G-kernel. (ii) An initial design selected via LHS with a number of points specified by a simple formula. (iii) The PS numerical search algorithm for finding \hat{x}_{o} that maximizes the acquisition function.

In general, (black-box) simulation implies that we do not know whether the problem has global optima besides local optima so we do not know whether EGO is desirable. Nevertheless, to be "safe", we may still apply EGO. In some simulations we might conjecture that the optima lie at the boundaries of the feasible areas, because the simulations have conflicting outputs. In practice, the users might restart an algorithm with different initial designs—possibly on a parallel computer—and select the result of the best restart.

We briefly discussed problems without binding constraints. Then we replaced the KKT conditions in our algorithm by the zero goal-gradient condition. In future research we may further investigate these problems.

We applied LOO-CV after the initial design, to investigate our combination of OK with a G-kernel. We could also apply LOO-CV after each iteration, to select the best combination of UK and a specific kernel (including an M-3/2 kernel and an M-5/2 kernel); however, we leave that generalization for future research.

In future research we may also investigate: (i) Choice of initial space-filling design. (ii) Customization of PS, and replacement of PS by an alternative numerical optimizer (e.g., fmincon). (iii) Comparison with additional alternative algorithms, including convergence proofs. (iv) High-dimensional optimization. (v) Parallel optimization. (v) Non-continuous inputs; e.g., integer and categorical inputs. (vi) Chanceconstrained problems in reliability engineering.

Acknowledgement

We thank the two anonymous reviewers for their very useful comments on a previous version. We also thank the following colleagues for their help with previous versions: Roymel Carpio (Universidade Federal do Rio de Janeiro (UFRJ), Brazil), Tony Pourmohamad (Genentech, Inc.), and Tianzeng Tao (Dalian University of Technology, China). Furthermore, we thank Dick den Hertog (University of Amsterdam) for clarifying the KKT conditions. This research was supported by the Flanders Artificial Intelligence Research Program (FLAIR).

References

- Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive blackbox functions.
 J. Global Optim. 13, 455–492 (1998)
- [2] Kushner, H.: A versatile stochastic model of a function of unknown and time-varying form. J. Math. Anal. Appl. 5, 150–167 (1962)
- [3] Mockus, J.: On Bayes methods for seeking an extremum. Avtomatika i Vychislitelnaja Technika 3, 53–62 (1972), in Russian
- [4] Zhigljavsky, A., Žilinskas, A.: Bi-objective decisions and partition-based algorithms in Bayesian global optimization. In: Bayesian and high-dimensional global optimization, pp. 41–88. SpringerBriefs in Optimization (2021)
- [5] Paulson, J.A., Tsay, C.: Bayesian optimization as a flexible and efficient design framework for sustainable process systems. https://arxiv.org/pdf/2401.16373 (2024)
- [6] Frazier, P.I.: A tutorial on Bayesian optimization. https://arxiv.org/pdf/1807.02811.pdf (2018)
- [7] Garnett, R.: Bayesian optimization. Cambridge University Press, United Kingdom (2023)
- [8] Wang, H., Yang, K.: Bayesian optimization. In: Many-Criteria Optimization and Decision Analysis, pp. 271–297, edited by D. Brockhoff, M. Emmerich, B. Naujoks, and R. Purshouse, Springer (2023)
- [9] Wang, X., Jin, Y., Schmitt, S., Olhofer, M.: Recent advances in Bayesian optimization. ACM Comput. Surv. 55(13s), Article 287, 36 pages, https://doi.org/10.1145/3582078 (2023)

- [10] Pourmohamad T., Lee, H.K.H.: Bayesian optimization via barrier functions. J. Comput. Graph. Stat. 31(1), 74–83 (2022)
- [11] Žilinskas, A., Calvin, J.: Bi-objective decision making in global optimization based on statistical models.
 J. Global Optim. 74, 599–609, https://doi.org/10.1007/s10898-018-0622-5 (2019)
- [12] Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press, United Kingdom (2009)
- [13] Aggarwal, C.C.: Linear algebra and optimization for machine learning. Springer (2020)
- [14] Carpio, R., Giordano, R.C., Secchi, A.R.: Enhanced surrogate assisted framework for constrained global optimization of expensive black-box functions. Comput. Chem. Eng. 118, 91-102, https://doi.org/10.1016/j.compchemeng.2018.06.027 (2018)
- [15] Gramacy, R.B., Gray, G.A., Le Digabel, S., Lee, H.K.H., Ranjan, P., Wells, G., Wild, S.M.: Modeling an augmented Lagrange for blackbox constrained optimization. Technometrics 58(1), 1–11 (2016) (Comments by several authors, and rejoinder by Gramacy et al. pp. 12–29)
- [16] Pourmohamad, T., Lee, H.K.H.: The statistical filter approach to constrained optimization. Technometrics 62(3), 303–312 (2020)
- [17] Letham, B., Karrer, B., Ottoni, G., Bakshy, E.: Constrained Bayesian optimization with noisy experiments. Bayesian Analysis 14(2), 495–519 (2019)
- [18] Liu, H., Xu, S., Chen, X., Wang, X., Ma, Q.: Constrained global optimization via a DIRECT-type constraint-handling technique and an adaptive metamodeling strategy. Struct. Multidiscip. O. 55(1), 155–177 (2017)
- [19] Zhang, Y., Zhang, X., Frazier, P.I.: Two-step lookahead Bayesian optimization with inequality constraints. In: 35th Conference on Neural Information Processing Systems, pp. 1-13, https: //proceedings.neurips.cc/paper/2021/file/685217557383cd194b4f10ae4b39eebf-Paper.pdf (2021)
- [20] Zhan, D., Xing, H.: Expected improvement for expensive optimization: a review. J. Global Optim. 78, 507-544, https://doi.org/10.1007/s10898-020-00923-x (2020)
- [21] Jones, D.R.: A taxonomy of global optimization algorithms based on response surfaces. J. Global Optim.
 21, 345–383 (2001)
- [22] Schonlau, M., Welch, W.J., Jones, D.R.: Global versus local search in constrained optimization of computer models. In: Lecture Notes-Monograph Series, pp. 11–25 (1998)

- [23] Kolman, B., Hill, D.R.: Elementary Linear Algebra with Applications, 9th edition. Pearson International Edition, Upper Saddle, New Jersey (2008)
- [24] Kleijnen, J.P.C., Mehdad, E.: Multivariate versus univariate Kriging metamodels for multi-response simulation models. Eur. J. Oper. Res. 236, 573–582 (2014)
- [25] Lim, C.Y., Wu, W-Y.: Conditions on which cokriging does not do better than kriging. J. of Multivariate Anal. 192, 105084 (2022)
- [26] Kleijnen, J.P.C.: Design and analysis of simulation experiments, second edition. Springer (2015)
- [27] Rasmussen, C., Williams, C.: Gaussian processes for machine learning. MIT Press, Cambridge, Massachusetts (2006)
- [28] Lophaven, S.N., Nielsen, H.B., Sondergaard, J.: DACE: a Matlab Kriging toolbox, version 2.0., IMM Technical University of Denmark, Kongens, Lyngby (2002)
- [29] Erickson, C.B., Ankenman, B.E., Plumlee, M., Sanchez, S.M.: Gradient-based criteria for sequential experiment design. Qual. Reliab. Eng. Int. 37(7), 3084–3107 (2021)
- [30] Kleijnen, J.P.C., van Beers, W.: Statistical tests for cross-validation of Kriging models. INFORMS J.Comput. 34(1), 607–621 (2022)
- [31] Ginsbourger, D., Schärer, C.: Fast calculation of Gaussian process multiple-fold cross-validation residuals and their covariances. J. Comput. Graph. Stat., https://doi.org/10.1080/10618600.2024.
 2353633 (2024)
- [32] Duvenaud, D.K.: Automatic model construction with Gaussian processes. Doctoral thesis, University of Cambridge, Pembroke, United Kingdom (2014)
- [33] Soleimani, M., Esmaeilbeigi, M., Cavoretto, R., De Rossi, A.: Analyzing the effects of various isotropic and anisotropic kernels on critical heat flux prediction using Gaussian process regression. Eng. Appl. Artif. Intel. 133, 108351, https://doi.org/10.1016/j.engappai.2024.108351 (2024)
- [34] Garud, S.S., Karimi, I.A., Kraft, M.: Design of computer experiments: a review. Comput. Chem. Eng. 106, 71–95 (2017)
- [35] Gómez, A.N., Pronzato, L., Rendas, M-J.: Incremental space-filling design based on coverings and spacings: improving upon low discrepancy sequences. Journal of Statistical Theory and Practice 15(4), 77 https://doi.org/10.1007/s42519-021-00210-2 (2021)

- [36] Jiang, P., Zhou, Q., Shao, X.: Surrogate model-based engineering design and optimization. Springer (2020)
- [37] Plumlee, M., Erickson, C.B., Ankenman, B.E., Lawrence, E.: Composite grid designs for adaptive computer experiments with fast inference. Biometrika 108(3), 749-755, https://doi.org/10.1093/ biomet/asaa084 (2021)
- [38] Tao, I., Zhao, G., Ren, S.: An efficient Kriging-based constrained optimization algorithm by global and local sampling in feasible region. J. Mech. Design 142(5), 051401-1 – 051401-15 (2020)
- [39] Loeppky, J.L., Sacks, J., Welch, W.: Choosing the sample size of a computer experiment: a practical guide. Technometrics 51(4), 366–376 (2009)
- [40] Wang, Z., Ierapetritou, M: Constrained optimization of black-box stochastic systems using a novel feasibility enhanced Kriging-based algorithm. Comput. Chem. Eng. 118, 210–223 (2018)
- [41] Pandita, P., Awalgaonkar, N., Bilionis, I., Panchal, J.: Learning arbitrary quantities of interest from expensive black-box functions through Bayesian sequential optimal design. https://arxiv.org/pdf/ 1912.07366 (2019)
- [42] Sóbester, A., Leary, S.J., Keane, A.J.: On the design of optimization strategies based on global response surface approximation models. J. Global Optim. 33(1), 31–59 (2005)
- [43] Miller, R.G.: Simultaneous statistical inference, revised second edition. Springer-Verlag, New York (1981) (original edition: McGraw-Hill, New York, 1966)
- [44] Picheny. V.: A stepwise uncertainty reduction approach to constrained global optimization. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, pp. 787-795 (2014)
- [45] Song, Z., Wang, H., Jin, Y.: A surrogate-assisted evolutionary framework with regions of interests-based data selection for expensive constrained optimization. IEEE T. Syst. Man Cy.-S. 53(10), 6268–6280 (2023)
- [46] Yang, Z., Qiu, H., Gao, L., Chen, L., Cai, X.: Constraint boundary pursuing-based surrogate-assisted differential evolution for expensive optimization problems with mixed constraints. Struct. Multidiscip. O., 28 pages (2023)

- [47] L'Ecuyer, P.: Random number generation with multiple streams for sequential and parallel computing.
 In: Proceedings of the 2015 Winter Simulation Conference, pp. 31–44, edited by L. Yilmaz, W.K.V.
 Chan, I. Moon, T.M.K. Roeder, C. Macal, and M.D. Rossetti (2015)
- [48] Kazemzadeh-Parsi, M.J.: A modified firefly algorithm for engineering design optimization problems.
 I.J.S.T.-T. Mech Eng 38(M2), 403–421 (2014)
- [49] Celik, Y., Kutucu, H.: Solving the tension/compression spring design problem by an improved firefly algorithm. https://ceur-ws.org/Vol-2255/paper2.pdf (2018)
- [50] Bagheri, S., Konen, W., Allmendinger, R., Branke, J., Deb, K., Fieldsend, J., Quagliarella, D., Sindhya, K.: Constraint handling in efficient global optimization. In: GECCO '17: Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, pp. 673–680, https://doi.org/10.1145/ 3071178.3071278 (2017)
- [51] Ray, T., Liew, K.M.: Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE T. Evolut. Comput. 7(4), 386–396 (2003)