Constrained Optimization in Simulation

Kleijnen, Jack; van Nieuwenhuyse, I.; van Beers, W.C.M.

Publication date:
2022

Document Version
Early version, also known as pre-print

No. 2022-020

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

By

Jack P.C. Kleijnen, Inneke van Nieuwenhuyse,
Wim C.M. van Beers

Revision of CentER DP 2021-031

August 2022

ISSN 0924-7815
ISSN 2213-9532

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

Jack P.C. Kleijnen
Department of Management, Tilburg School of Economics and Management (TiSEM), Tilburg University (TiU), Postbox 90153, 5000 LE Tilburg, Netherlands kleijnen@tilburguniversity.edu

Inneke Van Nieuwenhuyse
VCCM Core Lab, Flanders Make and Data Science Institute, Hasselt University, Belgium, inneke.vannieuwenhuyse@uhasselt.be

Wim van Beers
wimvanbeers@gmail.com

An important goal of simulation is optimization of the corresponding real system. We focus on simulation with multiple responses, selecting one response as the variable to be minimized while the remaining responses satisfy prespecified thresholds: so-called constrained optimization. We treat the simulation model as a black box. We assume that the simulation is computationally expensive; therefore, we use an inexpensive metamodel (emulator, surrogate) of the simulation model. A popular metamodel type is a Kriging or Gaussian process (GP) model (GP is also used in supervised learning). For optimization with a single response, this GP is used in efficient global optimization (EGO) (and also in Bayesian optimization, which is related to active learning).We develop an innovative EGO variant for constrained deterministic optimization where the optimal solution lies on one or more binding (input or output) constraints; therefore, we use the Karush-Kuhn-Tucker (KKT) conditions. We combine these conditions with the expected improvement (EI) criterion, which is popular in EGO. To evaluate the performance of our variant, we apply it to three popular examples (namely, one mathematical and two engineering design problems); these examples show promising numerical results when compared with other recent methods.

Key words: Kriging, efficient global optimization, Karush-Kuhn-Tucker conditions

---

JEL: C0, C1, C9, C15, C44

1. Introduction

Computer simulation (briefly, simulation) is a type of mathematical modeling that is popular in many scientific disciplines, in both research and practice; also see Calma et al. (2021). Simulationists often treat their model as a *black box*; i.e., they observe only the *input/output* (I/O) of the model. Mathematically speaking, a simulation model with vector input $\boldsymbol{x}$ and scalar response (output) w defines a complicated and implicit I/O (or transfer) function $w(\boldsymbol{x})$.

Specific simulation models may have different *goals*, such as optimization, prediction, sensitivity analysis (SA), and uncertainty quantification (UQ); see Kleijnen (2015, 2021). We focus on *constrained nonlinear single-objective optimization problems*. In general, this type of problem can be formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}} w_0(\boldsymbol{x}) \\ &w_{h'}(\boldsymbol{x}) \leq c_{h'} \ (h' = 1, \ldots, t-1) \\ &\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \end{aligned} \tag{1}$$

The $t > 1$ outputs w_h ($h = 0, ..., t-1$) are assumed to be deterministic. In our notation, w_0 is the *goal* variable, while $w_{h'}$ ($h' = 1, ..., t-1$) are the constrained outputs (with upper bounds $c_{h'}$). The input space is (in general) k-dimensional, and box constrained (i.e., it consists of continuous vectors $\boldsymbol{x} = \{x_1, ...x_k\}$; $\boldsymbol{l}$ and $\boldsymbol{u}$ are k-dimensional vectors with the lower and upper limits of the inputs). We assume that the simulation is *expensive*: i.e., it requires a lot of effort (either in terms of computation time, or cost) to obtain $w(\boldsymbol{x})$ for a given $\boldsymbol{x}$. In such settings, it is common to approximate the I/O functions through a simpler and explicit function, called a *metamodel* (or approximation, emulator, surrogate). A popular metamodel type, a.o. in *supervised learning*, is a *Kriging* model (named after Krige, a South-African mining engineer) or *Gaussian process* (GP) model: these GPs are used in the prominent *efficient global optimization* (EGO) method (see the seminal paper by Jones et al. (1998), which is a sequential statistical method that is also used in Bayesian optimization (BO) and is related to active learning). The term EGO occurs mainly in the Operations Research field,

while the other terms are more common in the Computer Science/Machine Learning field. EGO uses an *acquisition function* or *infill criterion* to iteratively select the next input combination to be simulated. The most popular criteria are expected improvement (EI) and probability of improvement (PI); a related criterion is the knowledge gradient in BO. The original publication by Jones et al. (1998) did not account for output constraints (only for box constraints on the inputs); later on, various authors have formulated variants on EGO (commonly using a suitable variant of the original EI criterion), e.g. for constrained optimization, multi-objective optimization, admissible or excursion sets, robust optimization, etc. (see the many references in Kleijnen (2015)).

We focus on problems as shown in Eq. (1), where at least one of constraints (either input or output constraints) is binding (or "active") in the optimum. Practical examples are abundant, for instance in the field of hydrology (Pourmohamad and Lee 2022), chemical engineering (Carpio et al. 2018), and aeronautical design (Tao et al. 2020, Bagheri et al. 2017). To solve this type of problem, various authors have proposed different methods. We develop an *innovative* method that combines the EI criterion with the *Karush-Kuhn-Tucker* (KKT, or briefly *Kuhn-Tucker*, KT) conditions to guide the search for better solutions. Even though these conditions are well known in nonlinear mathematical optimization (see Boyd (2009)), this paper presents (to the best of our knowledge) the only EGO variant that explicitly uses KKT conditions to guide the search process. Like other EGO variants, our method is a *heuristic* (i.e., it does not guarantee to find the global optimum). We also compare the results of our method and some alternative methods, found in the recent literature.

The remainder of this article is organised as follows: in Section 2, we discuss the literature on Kriging-based constrained simulation optimization. Section 3 gives an overview of the basics of Ordinary Kriging and Efficient Global Optimization. Section 4 discusses the KT conditions, while Section 5 explains the proposed KT-EGO algorithm. In Section 6, we present the results of our algorithm on three test problems: a toy problem originally put forward by Gramacy et al. (2016), and two mechanical engineering problems. Section 7 summarizes the conclusions.

2. Literature review: Kriging-based constrained simulation optimization

In this section, we place our work in the context of the literature on finding a global solution for an (expensive, deterministic) constrained nonlinear single-objective black-box optimization problem, as specified in Eq. (1). We focus on solution methods that use a metamodel; namely, a GP or Kriging metamodel, as is common in BO. Carpio et al. (2018) multiplies the PI for the goal variable with the product of the $(t-1)$ *probabilities of feasibility* (PFs) for the constraints. The method treats all output constraints as statistically independent. Similar methods are discussed in Bagheri et al. (2017) and Tran et al. (2019).

The algorithm presented in Tao et al. (2020) contains 3 phases, each having its own infill criterion: phase I tries to estimate the feasible region with high accuracy, phase II aims to further improve the global accuracy of the objective function in the feasible region, and phase III exploits the region around the optimal solution. Pourmohamad and Lee (2022) combines GPs with barrier function (BF) or interior point methods, which aim to ensure that the boundary of the constraint space (or feasible area) is never crossed. Our method, however, tries to estimate which input combinations lie near that boundary, using a cosine function to estimate to which extent the KKT conditions hold (evidently, some combinations may turn out to be infeasible when simulated). The BF methods outperform the *statistical filter* (SF) methods in Pourmohamad and Lee (2020) and the *augmented Lagrangian* (AL) methods in Gramacy et al. (2016), which also combine EGO with MathOpt. Picheny et al. (2016) further improved the AL approach.

3. Basics of Ordinary Kriging and EGO

Throughout this paper, we use the notation in Kleijnen (2021). For a given arbitrary output function h ($h=0...t-1$), Ordinary Kriging (OK) assumes the following (meta)model (we leave out the subscript h to improve readability):

$$y(\boldsymbol{x}) = \mu + M(\boldsymbol{x}) \tag{2}$$

where $\boldsymbol{x}$ is a combination of the k simulation inputs x_j ($j = 1, ..., k$), μ is the constant mean E[$y(\boldsymbol{x})$], and $M(\boldsymbol{x})$ is a realization of a zero-mean covariance stationary GP. This GP is fully defined by its mean function and its covariance function σ_M:

$$M(.) \sim \mathcal{GP}(0, \sigma_M(.,.))$$

The covariance function $\sigma_M(.,.)$ is assumed to exhibit spatial correlation: i.e., the outcomes $M(\boldsymbol{x_i})$ and $M(\boldsymbol{x_j})$ will tend to be similar when $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ are closer to each other in the input space. Note that we distinguish between the output function of the simulation model and the metamodel: strictly speaking, we may only write $y = w$ if and only if OK gives a fully *valid* metamodel of the underlying simulation model.

To estimate (or *train*) the metamodel, we need to have a set of I/O observations. So, let $\mathbf{X}$ denote the $n \times k$ matrix containing the input locations of n training sites, and let $\mathbf{w}$ denote the n-dimensional column vector containing the (deterministic) simulation output under study at these training sites $\mathbf{X}$. Assuming a valid metamodel, we can determine the *best linear unbiased predictor* (BLUP) $y(\boldsymbol{x}_*)$ at a new input location $\boldsymbol{x}_*$ using these I/O data $(\mathbf{X}, \mathbf{w})$. More specifically, this BLUP is a weighted average of the outcomes in $\mathbf{w}$:

$$y(\boldsymbol{x}_*) = \boldsymbol{\gamma'}\mathbf{w} \tag{3}$$

with

$$\boldsymbol{\gamma} = \boldsymbol{\Sigma}_M^{-1}\sigma_M(\boldsymbol{x}_i, \boldsymbol{x}_*)$$

In this expression, $\boldsymbol{\Sigma}_M$ is the $n \times n$ matrix containing the pairwise covariances between the outputs at the training locations, which we denote by:

$$\sigma_M(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = Cov[y(\boldsymbol{x}_i), y(\boldsymbol{x}_{i'})])$$

with $i, i^{'} = 1, ..., n$. Analogously, $\boldsymbol{\Sigma}_M(\boldsymbol{x}_*)$ is the $n \times 1$ vector containing the entries $\sigma_M(\boldsymbol{x}_i, \boldsymbol{x}_*) = Cov(y(\boldsymbol{x}_i), y(\boldsymbol{x}_*))$ for all training inputs $i = 1...n$. The pairwise covariance between two outputs is thus determined by their input locations. Sometimes, it is convenient to switch to the *correlation*

matrix $\mathbf{R} = (\rho_{i;i'})$, so $\mathbf{R} = \tau^{-2}\mathbf{\Sigma}_M$; analogously, $\boldsymbol{\rho}(\boldsymbol{x}_i, \boldsymbol{x}_*) = \tau^{-2}\boldsymbol{\sigma}_M(\boldsymbol{x}_i, \boldsymbol{x}_*)$ (with τ^2 the variance of the metamodel outputs y). There are several types of correlation functions; see (e.g.) Rasmussen and Williams (2006), pp. 80–104. In simulation, however, the most popular function is the *separable, anisotropic Gaussian correlation function*

$$\rho(\mathbf{d},\boldsymbol{\theta}) = \prod_{j=1}^{k} \exp\ \left(-\theta_j d_j^2\right) = \exp\ \left(-\sum_{j=1}^{k} \theta_j d_j^2\right) \text{ with } \theta_j \geq 0. \tag{4}$$

where $d_j = |x_{i;j} - x_{i';j}|$. As the value of θ_j approaches 0, the value of the correlation function increases to 1, implying that the y values are highly correlated (so their values vary slowly) along that dimension. Conversely, when θ_j approaches ∞, the correlation function drops to 0, and the y values may vary quickly along that dimension.

By definition, the resulting BLUP in Eq. (3) has minimum variance and is unbiased. Assuming a valid metamodel, the OK predictor at the novel input location $\boldsymbol{x}_*$ thus yields

$$y(\boldsymbol{x}_*) = \boldsymbol{\gamma}'\mathbf{w} = \mu + \mathbf{\Sigma}_M(\boldsymbol{x}_*)'\mathbf{\Sigma}_M^{-1}(\mathbf{w} - \mu\mathbf{1}_n). \tag{5}$$

and the *mean squared prediction error* (MSPE) of $y(\boldsymbol{x}_*)$ is

$$\text{MSPE}[y(\boldsymbol{x}_*)] = \tau^2 - \mathbf{\Sigma}_M(\boldsymbol{x}_*)'\mathbf{\Sigma}_M^{-1}\sigma_M(\boldsymbol{x}_i, \boldsymbol{x}_*) + \frac{[1 - \mathbf{1}_n'\mathbf{\Sigma}_M^{-1}\mathbf{\Sigma}_M(\boldsymbol{x}_*)]^2}{\mathbf{1}_n'\mathbf{\Sigma}_M^{-1}\mathbf{1}_n}. \tag{6}$$

When the metamodel is valid (so $y(\boldsymbol{x}_*)$ is unbiased), MSPE$[y(\boldsymbol{x}_*)]$= $s^2[y(\boldsymbol{x}_*)]$.

We let $\boldsymbol{\psi}$ denote the vector with the $(2 + k)$ *Kriging (hyper)parameters* $(\mu, \tau^2, \theta_1, ..., \theta_k)'$. In practice, this $\boldsymbol{\psi}$ is unknown, and is commonly estimated through its *maximum likelihood estimator* (MLE), $\widehat{\boldsymbol{\psi}} = \widehat{\boldsymbol{\psi}}(\mathbf{X}, \mathbf{w})$. To obtain estimates of $y(\boldsymbol{x}_*, \widehat{\boldsymbol{\psi}})$ and $s^2[y\boldsymbol{x}_*, \widehat{\boldsymbol{\psi}})]$, we *plug* $\widehat{\boldsymbol{\psi}}$ into (5) and (6). We ignore the bias caused by the nonlinearity of $\widehat{y}(\boldsymbol{x}_*, \widehat{\boldsymbol{\psi}})$, and the bias in $\widehat{s}^2[\widehat{y}(\boldsymbol{x}_*, \widehat{\boldsymbol{\psi}})]$ (which is known to underestimate the true variance; to obtain an unbiased variance estimator, bootstrapping could be used, see Kleijnen and van Beers (2021). Obviously, $\boldsymbol{x}_*$ in $\widehat{y}(\boldsymbol{x}_*, \widehat{\boldsymbol{\psi}}_h)$ affects only $\widehat{\mathbf{\Sigma}}_M(\boldsymbol{x}_*)$—not $\widehat{\mu}$ and $\widehat{\mathbf{\Sigma}}_M^{-1}$, which are determined by $(\mathbf{X}, \mathbf{w})$. So, we may rewrite (5) as

$$\widehat{y}(\boldsymbol{x}_*, \widehat{\boldsymbol{\psi}}) = \widehat{\mu} + \widehat{\mathbf{\Sigma}}_M(\boldsymbol{x}_*)'\widehat{\mathbf{c}} \text{ with } \widehat{\mathbf{c}} = \widehat{\mathbf{\Sigma}}_M^{-1}(\mathbf{w} - \widehat{\mu}\mathbf{1}_n), \tag{7}$$

which we shall use in the next subsection. Note that we estimate a separate OK model per simulation output, so we obtain t such models; we do not use *multivariate* Kriging (also called *co-Kriging*; see, for instance, Gramacy et al. (2016), Pourmohamad and Lee (2020), Sadoughi et al. (2018)). Our approach simplifies the analysis; moreover Kleijnen (2014) reports that both approaches may give similar results. It is well known that OK yields an *exact* interpolator: i.e., at the already observed input locations $\boldsymbol{x}_i$ $(i = 1...n)$, $\widehat{s}^2[\widehat{y}(\boldsymbol{x}_i, \widehat{\boldsymbol{\psi}}] = 0$.

4. KT-conditions in constrained optimization

In constrained nonlinear mathematical optimization, the *KT conditions* are first-derivative (or first-order) necessary conditions for a solution x_*) to be optimal, provided that some regularity conditions are satisfied (these conditions also involve Lagrangian multipliers; see again Gramacy et al. (2016)). Assuming that all output functions of our problem (see Eq. (1)) are differentiable, we formalize the KT conditions as follows:

$$-\nabla_0(\boldsymbol{x}_*) = \sum_{h'' \in A_\lambda(\boldsymbol{x}_*)} \lambda_{h''}(\boldsymbol{x}_*)\nabla_{h''}(\boldsymbol{x}_*) + \sum_{g'' \in A_\mu(\boldsymbol{x}_*)} \mu_{g''}(\boldsymbol{x}_*)\nabla_{g''}(\boldsymbol{x}_*)$$

$$\text{with } \lambda_{h''}(\boldsymbol{x}_*) \geq 0 \text{ and } \mu_{g''}(\boldsymbol{x}_*) \geq 0. \tag{8}$$

In this expression, the k-dimensional vector $\nabla_0(x_*) = (\partial w_0/\partial x_1, ..., \partial w_0/\partial x_k)'|_{\boldsymbol{x}_*}$ denotes the *gradient* of the goal function $w_0(x)$ at point x_*. Likewise, $\nabla_{h''}(x)$ denotes the k-dimensional vector with the gradient of the *binding output constraint* function $w_{h''}$ at x, while the notation $\nabla_{g''}$ is the k-dimensional vector containing the gradient of the *binding input constraint* g''. The notation $A_\lambda(\boldsymbol{x}_*)$ refers to the set with all indices (or subscripts) h'' of the binding output constraints, while $A_\mu(\boldsymbol{x}_*)$ refers to the set with the indices g'' of the binding input constraints at x_*. Note that the KT conditions imply that at least 1 constraint (output or input constraint) is binding. Obviously, gradients of *non-binding* constraints play no role in Eq. (8). If min in Eq.(1) is replaced by max, $-\nabla_0(x_*)$ in Eq.(8) is replaced by $\nabla_0(x_*)$.

Eq.(8) can be interpreted as a linear regression model, where the explanatory variables are given by the matrix $\boldsymbol{\Delta} = (\nabla_{h''}, \nabla_{g''})$, and the dependent variable is given by $-\nabla_0$ (we omit the argument

($\boldsymbol{x}_*$) to avoid cumbersome notation). This gives the LS estimator $\widetilde{\boldsymbol{\nu}}$ (in the remainder of this text, the notation $\widetilde{(.)}$ denotes a *LS estimator*, while the notation $\widehat{(.)}$ denotes a *MLE estimator*):

$$\widetilde{\boldsymbol{\nu}} = (\widetilde{\boldsymbol{\lambda}}', \widetilde{\boldsymbol{\mu}}')' = (\boldsymbol{\Delta}'\boldsymbol{\Delta})^{-1}\boldsymbol{\Delta}'(-\nabla_0) \tag{9}$$

We estimate the gradient vectors in (8) and (9) through their MLE estimators. Applying basic calculus, we can prove that (7) gives (also see Chen et al. (2021)):

$$\nabla[\widehat{y}(\boldsymbol{x}_*, \widehat{\boldsymbol{\psi}})] = \nabla[\widehat{\boldsymbol{\Sigma}}_M(\boldsymbol{x}_*)'\widehat{\mathbf{c}}]. \tag{10}$$

Assuming the Gaussian covariance function, for instance, this gradient has the following k elements where $\widehat{c}_j$ is element j (j = 1, .., k) of $\widehat{\mathbf{c}}$:

$$\frac{\partial \widehat{y}(\boldsymbol{x}_*, \widehat{\boldsymbol{\psi}})}{\partial x_{*;j}} = -2\widehat{\tau}^2\widehat{\theta}_j\{\Sigma_{i=1}^n \widehat{c}_i(x_{*;j} - x_{i:j})\exp[\Sigma_{j'=1}^k - \widehat{\theta}_{j'}(x_{*;j'} - x_{i:j'})^2]\} \tag{11}$$

These partial derivatives are readily calculated in DACE. We then obtain the following LS estimator for the goal gradient:

$$-\widetilde{\nabla}_0(\boldsymbol{x}_*) = \sum_{h'' \in A_\lambda(\boldsymbol{x}_*)} \widetilde{\lambda}_{h''}(\boldsymbol{x}_*)\widehat{\nabla}_{h''}(\boldsymbol{x}_*) + \sum_{g'' \in A_\mu(\boldsymbol{x}_*)} \widetilde{\mu}_{g''}(\boldsymbol{x}_*)\widehat{\nabla}_{g''}(\boldsymbol{x}_*)$$

$$\text{with } \widetilde{\lambda}_{h''}(\boldsymbol{x}_*) \geq 0 \text{ and } \widetilde{\mu}_{g''}(\boldsymbol{x}_*) \geq 0. \tag{12}$$

To quantify the *validity* of this LS model, we use $\cos[-\widehat{\nabla}_0(\boldsymbol{x}_*),\ -\widetilde{\nabla}_0(\boldsymbol{x}_*)]$) (see, e.g., Kolman (2008), p. 73), which equals:

$$\cos[\widehat{\nabla}_0(\boldsymbol{x}_*), \widetilde{\nabla}_0(\boldsymbol{x}_*)] = \frac{\widehat{\nabla}_0(\boldsymbol{x}_*) \bullet \widetilde{\nabla}_0(\boldsymbol{x}_*)}{||\widehat{\nabla}_0(\boldsymbol{x}_*)|| \times ||\widetilde{\nabla}_0(\boldsymbol{x}_*)||}. \tag{13}$$

Ideally, the two vectors coincide, yielding a cosine equal to 1 (implying that the LS model gives a perfect fit, or the coefficient of determination—denoted by R^2—equals 1). In general, though, this will seldom be the case; in our algorithm, we require only that these two vectors point in the *same direction* if $\boldsymbol{x}_*$ lies *near* a stationary point; i.e., we require that $0 \leq \cos[\widehat{\nabla}_0(\boldsymbol{x}_*),\ \widetilde{\nabla}_0(\boldsymbol{x}_*)] \leq 1$ (obviously, values closer to 1 are preferable).

Algorithm 1 KT-EGO algorithm

Initialize $\epsilon = 0.001$, $\alpha_{LB} = 0.01$, $iter = 0$

Step 1: Initial design: Use LHS to sample the $n \times k$ matrix $\mathbf{X}$, containing the n initial design combinations. Use $\mathbf{X}$ as input for the given simulation model, to obtain the t output vectors $\mathbf{w}_h$ ($h = 0, ..., t-1$).

Step 2: Initialize new iteration: Set $\alpha = 0.2$, $iter = iter + 1$.

Step 3: Determine current best goal value: Determine the best goal value ($w_{0;\,\min}$) among the feasible simulated points: $w_{0;\,\min} = \min_{1 \le i \le n}(w_0(\widehat{\boldsymbol{x}}_i))$, $\forall i : w_{h'}(\widehat{\boldsymbol{x}}_i) \le c_{h'}$ $(h' = 1, ..., t-1)$. If none of the simulated points is feasible, set $w_{0;\,\min} = \infty$.

Step 4: Estimate ordinary Kriging models: Use the simulation I/O data $(\mathbf{X}, \mathbf{w}_h)$ ($h = 0, ...,$ $t-1$) to estimate t univariate OK models, yielding the MLE of the OK parameters $\widehat{\boldsymbol{\psi}}_h = \widehat{\boldsymbol{\psi}}(\mathbf{X}, \mathbf{w}_h)$ ($h = 0, ..., t-1$).

Step 5: Search for infill point: Estimate the infill point $\widehat{\boldsymbol{x}}_o$, i.e. the point that maximizes the infill criterion $\widehat{\mathrm{EI}}_0(\boldsymbol{x}) \times \cos[\widehat{\nabla}_0(\boldsymbol{x}), \widetilde{\nabla}_0(\boldsymbol{x})]$, with $\widehat{\mathrm{EI}}_0(\boldsymbol{x})$ as in Eq.(15), and $\cos[\widehat{\nabla}_0(\boldsymbol{x}), \widetilde{\nabla}_0(\boldsymbol{x})]$ given by Eq.(13).

Step 6: Check end of iteration: If the optimization in Step 5 found a feasible infill point $\widehat{\boldsymbol{x}}_o$ with $\widehat{\mathrm{EI}}_0(\boldsymbol{x_o}) > \varepsilon \times w_{0;\,\min}$, go to Step 7; otherwise, go to Step 8.

Step 7: Update I/O data: Use $\widehat{\boldsymbol{x}}_o$ as input for the simulation model; update the matrix $\mathbf{X}$ and the t output vectors $\mathbf{w}_h$ ($h = 0, ..., t-1$) with the new I/O information. Return to Step 2.

Step 8: Try to reduce α: If $\alpha/2 \ge \alpha_{LB}$, then replace α by $\alpha/2$ and return to Step 5. Otherwise, go to Step 9.

Step 9: Estimate final optimum: Optimize the Kriging estimate $\widehat{y}_0$ directly, subject to the requirement that the upper bounds of the 90% confidence intervals on the $t-1$ constraint estimates need to be negative $(\widehat{y}_{h'}(\boldsymbol{x}) + z_{1-\alpha/2} * s[\widehat{y}_{h'}(\boldsymbol{x})]$, with $\alpha = 0.2$, $h\prime = 1...t-1)$. Simulate the input vector found ($\mathbf{x_{est}}$). If $\mathbf{x_{est}}$ is feasible, the estimated optimum goal value equals $min(w_{0;\,\min}, w_{0;\mathrm{est}})$ (with $w_{0;\,\min}$ as determined in Step 3). Otherwise, this value equals $w_{0;\mathrm{min}}$.

5. Efficient global optimization and Kuhn-Tucker conditions: KT-EGO algorithm

In this section, we present a variant of EGO that takes into account the Kuhn-Tucker conditions. The steps of the algorithm are outlined in Algorithm 1.

In Step 1, we generate the initial design matrix $\mathbf{X}$, containing n combinations of the k simulation input combinations, and use it as input in the simulation model to obtain the n (t-variate) simulation outputs $\mathbf{w}_{n;h}$ ($h = 0,..., t-1$). A popular choice—based on Loeppky et al. (2009)—is $n = 10k$; yet, in our experiments, we opt to use the smaller design size proposed in Tao et al. (2020), which sets $n = min(5k, (k+1)(k+2)/2)$ for $k \leq 6$, and $n = 5k$ otherwise. We opt for an LHS design, as this is the most commonly used space-filling design in the Kriging literature. Step 2 then initializes the values of α (which determines the confidence interval width in Eq.(16), see Step 5 below), and the iteration counter at the start of a new iteration. Step 3 determines the current best point among the (feasible) simulated points: we let $w_{0;\,\min}$ denote its goal output. If none of the current simulated points is feasible, we set $w_{0;\,\min} = \infty$. In Step 4, we use the simulation I/O data $(\mathbf{X}, \mathbf{w}_h)$, to obtain the estimated parameters of the t univariate OK metamodels: $\widehat{\boldsymbol{\psi}}_h = (\widehat{\mu}_h, \widehat{\tau}_h^2, \widehat{\theta}_{1;h}, ..., \widehat{\theta}_{k;h})'$. In our experiments, we used Gaussian correlation functions. Step 5 is the key step of our algorithm: here, we select the next point to be simulated ($\boldsymbol{x}_{\text{o}}$; also referred to as the *infill point*) by maximizing the following novel infill criterion:

$$\widehat{\text{EI}}_{\text{KT}}(\boldsymbol{x}) = \widehat{\text{EI}}_0(\boldsymbol{x}) \times \cos[\widehat{\nabla}_0(\boldsymbol{x}), \widetilde{\nabla}_0(\boldsymbol{x})]. \tag{14}$$

The first factor refers to the well-known expected improvement criterion, which can be estimated from the Kriging information (see, e.g., Jones et al. (1998)):

$$\widehat{\text{EI}}_0(\boldsymbol{x}) = (w_{0;min} - \widehat{y}(\boldsymbol{x}))\, \Phi\left(\frac{w_{0;min} - \widehat{y}(\boldsymbol{x})}{s[\widehat{y}(\boldsymbol{x})]}\right) + s[\widehat{y}(\boldsymbol{x})]\phi\left(\frac{w_{0;min} - \widehat{y}(\boldsymbol{x})}{s[\widehat{y}(\boldsymbol{x})]}\right). \tag{15}$$

This criterion balances *local* search (*or exploitation*, in regions with high uncertainty $s[\widehat{y}(\boldsymbol{x})]$) and *global* search (*exploration*, in regions with promising values for $\widehat{y}(\boldsymbol{x})$). The second factor quantifies

the validity of the KKT conditions (see Eq. (13)). To determine this factor, we need to estimate which are the *binding output constraints* in any novel point (detecting binding input constraints is trivial). We estimate an output constraint $h\prime$ to be binding if the following condition holds:

$$\frac{|\widehat{y}_{h'}(\boldsymbol{x}) - c_{h'}|}{s[\widehat{y}_{h'}(\boldsymbol{x})]} \leq z_{1-\alpha/2} \tag{16}$$

where z is the $(1-\alpha/2)$ percentile of the standard normal probability distribution (so $z_{1-\alpha/2} = \Phi^{-1}(1-\alpha/2)$), and α is a prespecified small probability (e.g., in our experiments, we initialize α equal to 0.20, which gives $z_{0.90} \approx 1.2816$). So, essentially, we estimate a constraint to be binding at an arbitrary point $\boldsymbol{x}$ if the two-sided $(1-\alpha)$ confidence interval on the predictor includes the right-hand side of the constraint (see the problem formulation in Eq.(1)). Note that $[c_{h'} - \widehat{y}_{h'}(\boldsymbol{x})]/s[\widehat{y}_{h'}(\boldsymbol{x})]$ (Studentized slack of any arbitrary output constraint h') is *scale-free* (so each constraint may have its own measurement unit).

Maximizing the criterion in Eq. (14) is a difficult task, as its response surface over the search space is usually highly multimodal. To find $\boldsymbol{x}_{\text{o}}$, we applied MATLAB's pattern search (hereafter referred to as *PS*); evidently, other heuristic search algorithms can also be used. To reduce the probability of PS getting trapped in a local optimum, we restart the algorithm multiple times, with different starting points. Step 6 checks whether the optimal infill point returned by PS has a sufficiently high value for $\widehat{\text{EI}}_0$ (in our experiments, we require it to be strictly larger than 0.1% of the current best goal value); if so, the infill point is simulated, and its I/O data are used to update $(\mathbf{X}, \mathbf{w})$ and re-estimate the Kriging models (Step 7), after which the algorithm returns to Step 2 to start a new iteration. The simulation in Step 7 may reveal that the suggested infill point is actually infeasible; although it doesn't lead to an update of $w_{0;\,\text{min}}$ in that case (Step 3 in the new iteration), such point still provides useful information for the re-estimation of the Kriging models (Step 4). It may happen, though, that Step 6 reveals that no suitable infill point has been found (e.g., because the optimization in Step 5 was unable to find any point with binding constraints, or because the expected $\widehat{\text{EI}}_0(\boldsymbol{x})$ in all these points is too small to reach the required threshold). In that case, we decrease the value of α by a factor 2 (Step 8): this increases the width of the confidence

intervals in Eq.(16), causing the search to classify more points as potential infill points (i.e., having at least one binding constraint). The algorithm continues to iteratively add infill points, until a stop criterion is met. In our experiments, this happens when the algorithm doesn't find a suitable infill point in Step 6, and further decreasing α would result in α violating a (user-defined) lower bound α_{LB} (in our experiments, we set $\alpha_{LB} = 0.01$). Once the stopping criterion is met, the algorithm proceeds to Step 9: here, the Kriging estimate of the goal function $\widehat{y_0}$ is optimized directly, subject to the requirement that the upper bounds of the 90% confidence intervals on the $t-1$ constraint estimates need to be negative ($\widehat{y}_{h'}(\boldsymbol{x}) + z_{1-\alpha/2} * s[\widehat{y}_{h'}(\boldsymbol{x})]$, with $\alpha = 0.2$, $h\prime = 1...t-1$). Again, this can be done using pattern search, or another heuristic optimization approach. We denote the resulting estimated optimum combination by $\mathbf{x_{est}}$. Simulating this combination shows whether this estimated optimum is feasible: if so, the final optimum goal value returned by the algorithm equals $min(w_{0;\,\min}, w_{0;\mathrm{est}})$ (with $w_{0;\,\min}$ as determined in Step 3). Otherwise, it equals $w_{0;\mathrm{min}}$. Evidently, in this final optimum, the slack should be virtually zero for at least one constraint.

6. Numerical results

In this section, we discuss the results obtained when applying our KT-EGO algorithm to a toy problem originally put forward by Gramacy et al. (2016) (see Section 6.1), and two well-known mechanical engineering problems, i.e., the tension-compression spring problem (Section 6.2) and the I-beam design problem (Section 6.3). The mechanical engineering field offers many popular examples of constrained optimization in deterministic simulation; we chose these two examples since the I/O functions are explicitly known, and they have a small number of (continuous) inputs.

We compare the performance of our KT-EGO algorithm (on each of these problems) with the performance of alternative algorithms published in the literature. To that end, we run each problem for a given number of *macroreplications* (each macroreplication starts with a different initial LHS design; that design is then used by each of the algorithms, to achieve a fair comparison).

6.1. Toy problem

This toy problem was originally put forward in Gramacy et al. (2016), and has also been used as a test problem in Pourmohamad and Lee (2020, 2022). It is formalized as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} w_0(\boldsymbol{x}) &= x_1 + x_2 \\ w_1(\boldsymbol{x}) &= \frac{3}{2} - x_1 - 2x_2 - \frac{1}{2}\sin[2\pi(x_1^2 - 2x_2)] \leq 0 \\ w_2(\boldsymbol{x}) &= -\frac{3}{2} + x_1^2 + x_2^2 \leq 0 \\ 0 \leq x_j &\leq 1 \ (j = 1, 2). \end{aligned} \tag{17}$$

Fig. 1 displays the feasible region of this problem by the dotted area between $w_1(\boldsymbol{x}) = 0$ (highly nonlinear and nonconvex curve) and $w_2(\boldsymbol{x}) = 0$ (circle with radius $\sqrt{3/2}$, partly displayed in upper right corner). It also displays three iso-curves (red dashed lines) for w_0 (all input locations on a given iso-curve have the same w_0). The points A through E are all stationary points; they satisfy the KKT conditions. Among these, points D and E are local *maxima*, while points A through C are local *minima*. More specifically, $\boldsymbol{x}_{\mathrm{A}} \approx (0.1954, 0.4044)'$ with $w_0(\boldsymbol{x}_{\mathrm{A}}) \approx 0.5998$, $\boldsymbol{x}_{\mathrm{B}} \approx (0.7197, 0.1411)'$ with $w_0(\boldsymbol{x}_{\mathrm{B}}) \approx 0.8609$, and $\boldsymbol{x}_{\mathrm{C}}{=}(0, 0.75)'$ with $w_0(\boldsymbol{x}_{\mathrm{C}}) = 0.75$. So, the global optimum is $w_0(\boldsymbol{x}_{\mathrm{A}})$.

We run our algorithm starting from an initial LHS design of 6 points (as $k = 2$, $n = (k+1)(k+2)/2$; see Section 5). In each iteration i, the pattern search is restarted from $n_s = 20$ starting points, consisting of the infill points obtained in the n_s most recent iterations; when less than n_s infill points are available (i.e., when $i - 1 < n_s$), the remaining restarting points are generated by means of an LHS design.

Fig. 2a illustrates the search process of the algorithm in the first macroreplication. The left pane shows the infill points selected during the search, with the numbers indicating the corresponding iteration (where 0 refers to the best point present in the initial design; feasible infill points are shown as a full circle, infeasible points as an empty circle). The right pane shows the evolution of the best goal value ($w_{0;min}$) found during the search in terms of the total number of expensive evaluations performed ($n_{0;min}$, which includes the initial design evaluations). Fig. 2b shows the

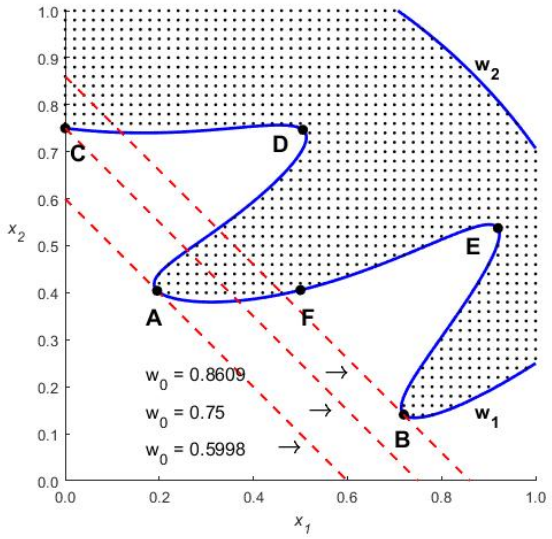


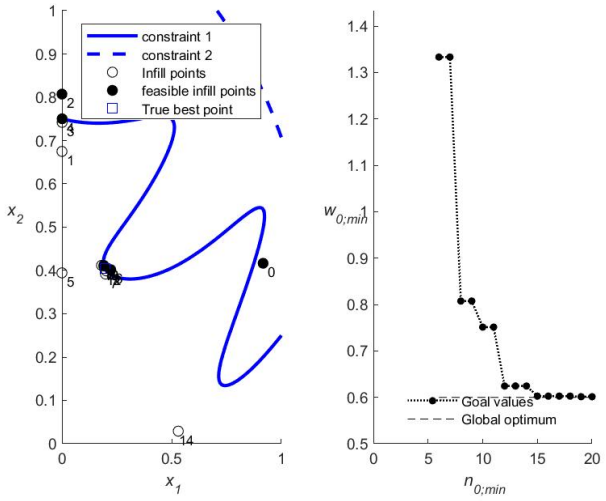
Figure 1 Toy example with inputs x_1 and x_2, goal output w_0, constraints w_1 and w_2, and special points A through F.

same, for macroreplication 2. As discussed above, the algorithm stops when it is unable to find a next infill point that has a sufficiently high value for $\widehat{\mathrm{EI}}_0(\boldsymbol{x})$, and $\alpha = 0.0125$, so it cannot further be decreased; see Algorithm 1.

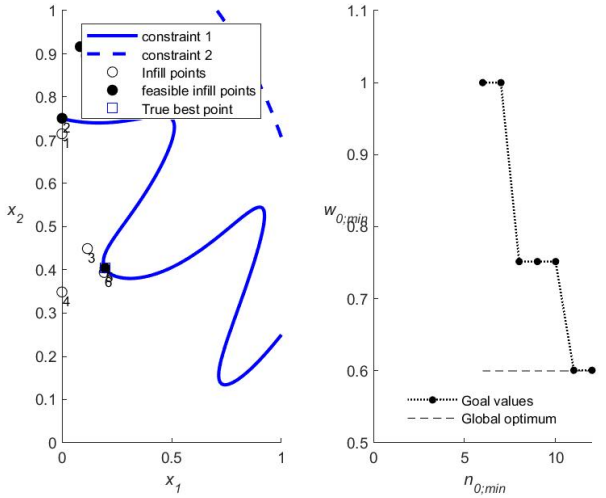
It is clear that the algorithm focuses quickly on the area of the global optimum. We performed $m = 50$ replications of our algorithm on this toy problem; each macroreplication samples a statistically independent initial LHS design. Each replication uses the same initial value (0.20) for α. Fig. 3 shows the boxplot of the best found goal values at the end of the 50 macroreplications (left pane), and the number of expensive evaluations performed by the algorithm (right pane).

Obviously, the final best found goal value cannot be lower than the true global minimum $w_0(\boldsymbol{x}_\mathrm{A}) \approx 0.5998$). Our algorithm obtains an estimated mean objective value of 0.6100, with a median of 0.6000 (see also Table 1). The boxplot for the number of expensive evaluations shows a mean number equal to 18.58, with an estimated median of 18. Altogether, KT-EGO gives a quite precise estimator of the global minimum, within a very small number of iterations.

Finally, Fig. 4 and Table 1 compare the numerical results of our KT_EGO algorithm with those obtained with three recent alternatives for constrained Bayesian optimization: (i) Pourmohamad



(a) Macroreplication 1



(b) Macroreplication 2

Figure 2 Illustration of the infill points selected in the different iterations of the algorithm (left pane) and the evolution of the best goal value obtained in terms of the number of expensive evaluations ($w_{0;min}$ in terms of $n_{0;min}$; right pane) in macroreplications 1 and 2 of the toy problem.

and Lee (2022), (ii) Carpio et al. (2018), and (iii) Tao et al. (2020). While Tao et al. (2020) uses an initial design of $n = 6$ points, Pourmohamad and Lee (2022) and Carpio et al. (2018) both use $n = 20$; to facilitate comparison, we thus also add the results for KT_EGO with $n = 20$ to the figure.

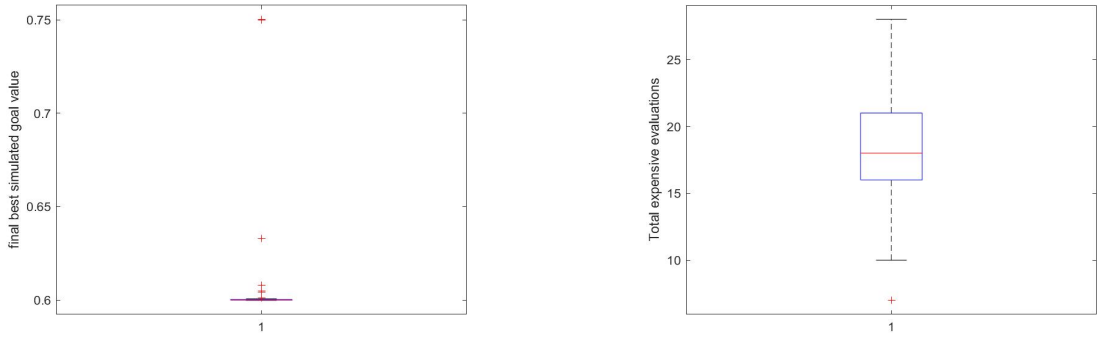


Figure 3 Boxplots of the final best goal values found (left pane) and the number of expensive evaluations performed (right pane), for 50 macroreplications, for the toy problem.

Table 1 Comparison of numerical results for the toy problem

Algorithm	**Best goal value**			**Nr of expensive evaluations**		
	Mean	Median	Std	Mean	Median	Std
$n = 20$						
KT-EGO	0.6012	0.6000	0.0077	29.84	29.00	2.8881
Pourmohamad and Lee (2022)	0.6036	0.6030	0.0028	60	60	0
Carpio et al. (2018)	0.6027	0.5998	0.0046	40.46	40.5	5.0314
$n = 6$						
KT-EGO	0.6100	0.6000	0.0361	18.58	18.00	4.4175
EKCO, Tao et al. (2020)*	0.5998	0.5998	4.24E-05	32.12	32	3.4561

* The algorithm by Tao et al. (2020) reports the Kriging output of the estimated Kriging optimum (which we refer to as $\mathbf{x_{est}}$, Step 9 of our algorithm) as the final optimum.

The *barrier function* (BF) methods, presented in Pourmohamad and Lee (2022), have proven to outperform the statistical filter (SF) methods in Pourmohamad and Lee (2020), and the augmented Lagrangian (AL) methods in Gramacy et al. (2016). These BF, SF, and AL methods combine EGO with mathematical optimization, as KT-EGO does. More precisely, BF methods try to optimize the goal function while taking care that the boundary of the feasible area is never crossed. Three BF methods were put forward in Pourmohamad and Lee (2022); at our request, Pourmohamad

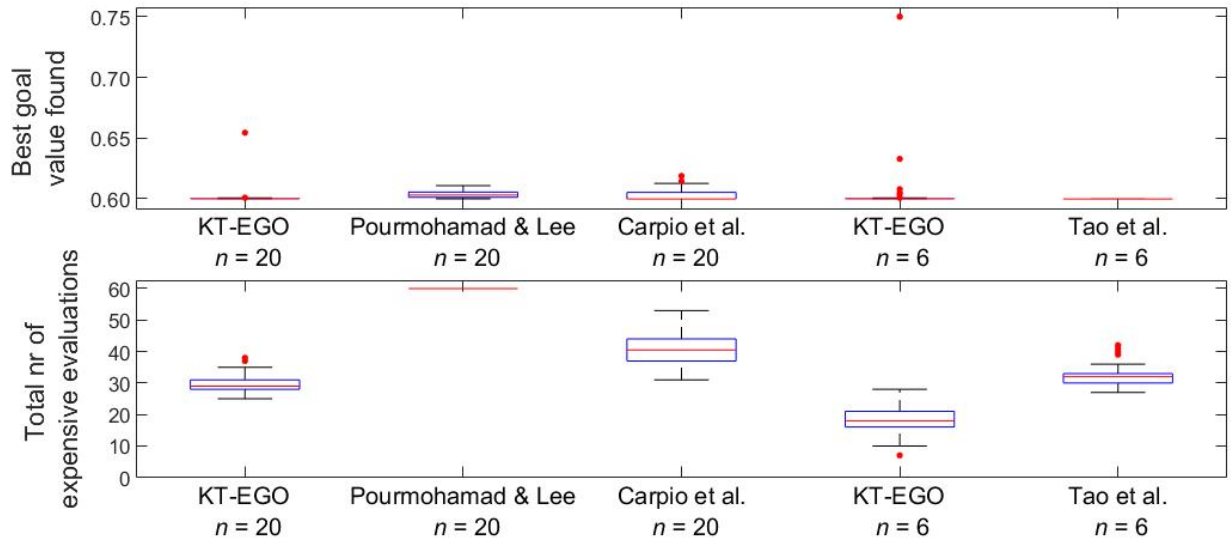


Figure 4 Comparison of final best goal values found, and total number of expensive evaluations performed, for KT_EGO (with $n = 6$ and $n = 20$ initial design points), and the algorithms by Pourmohamad and Lee (2022; $n = 20$), Tao et al. (2020; $n = 6$), and Carpio et al. (2018; $n = 20$).

applied the best of these to the toy problem. His experiment uses an initial LHS design with 20 points (as KT-EGO does) and 40 additional iterations, in 50 macroreplications. Fig. 4 shows that our KT-EGO algorithm is able to obtain a mean and median outcome for the best goal value that is slightly better than the one obtained by Pourmohamad and Lee (2022). The $(25\%, 75\%)$ percentiles also compare favorably. KT-EGO obtains this outcome within a substantially smaller number of expensive evaluations. We conclude that our method shows a favorable trade-off between quality and speed, compared to the one put forward by Pourmohamad and Lee (2022).

The algorithm of Carpio et al. (2018) uses a dynamic stop criterion, based on the constrained probability of improvement PI_c: the algorithm stops when PI_c reaches the threshold value 10^{-3} or 10^{-4}. At our request, Carpio applied this method to the toy problem, using 50 macroreplications, with threshold $PI_c = 10^{-4}$. This resulted in a median of 40.50 expensive evaluations. Our method is slightly better w.r.t. the mean goal value, and similar w.r.t. the median; it again compares favorably in terms of speed.

The 3-phase EKCO algorithm by Tao et al. (2020) uses $n = (k+1)(k+2)/2 = 6$ initial design points. At our request, Tao applied this method to the toy problem, using 50 macroreplications.

Note that the algorithm by Tao et al. (2020) reports the Kriging output of the estimated Kriging optimum (which we refer to as $\mathbf{x_{est}}$, Step 9 of our algorithm) as the final optimum; the results are thus "optimistic", as the algorithm does not simulate this optimum to check its feasibility. The EKCO algorithm gives a median goal value equal to 0.59979 (the scale of Fig. 4 is such that it does not show the variation in this final goal value, which is very small). Yet, the estimated median number of expensive evaluations is 32, which is significantly higher than the median for KT-EGO. Note that the smaller design size $n = 6$ leads to substantial decrease in the number of expensive evaluations required for KT-EGO, without significantly impacting the quality of the solution.

In summary, we can conclude that KT-EGO is very efficient (as measured by number of expensive evaluations required) and has similar effectiveness (as shown by the final best goal value found) when compared with the three alternative methods —albeit with some outliers. Some outliers correspond with final solutions that end near point C (which does satisfy the KT conditions, so it is a stationary point, but not a global optimum). Furthermore, in some macroreplications, our method rejects solutions near point A because it estimates these solutions to be infeasible; consequently, it continues its search and ends with solutions that are further off, resulting again in an "outlier". Altogether, we expect that our method quickly gives a solution close to the true optimum, but occasionally it does not.

6.2. Tension-compression spring design problem

The spring design problem (see Fig. 5) is discussed in (a.o.) Kazemzadeh-Parsi (2014), and Tao et al. (2020). In this problem, the inputs are the number of active coils N (x_1), the wire diameter d (x_2), and the mean coil diameter D (x_3). The goal is to minimize the weight of the spring (w_0), subject to constraints on the minimum deflection of the spring caused by the axial loading (w_1), the maximum shear stress (w_2), the surge frequency (w_3), and the outside diameter of the spring (w_4), while keeping the inputs within the box constraints.

We base our mathematical model on Tao et al. (2020) (Supplement, Problem 16):

$$\min_{\boldsymbol{x}} w_0(\boldsymbol{x}) = (x_1 + 2)x_2x_3^2$$

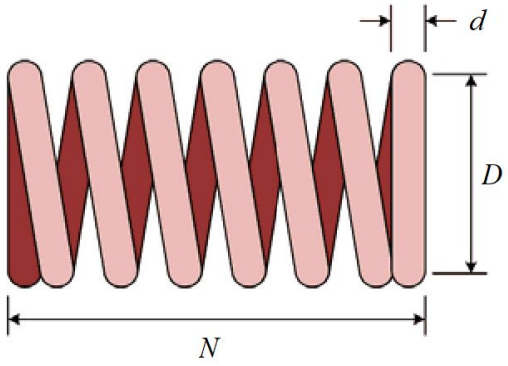


Figure 5 Tension-compression spring design problem.

$$
\begin{aligned}
w_1(\boldsymbol{x}) &= 1 - \frac{x_2^3 x_1}{71,875 x_3^4} \leq 0 \\
w_2(\boldsymbol{x}) &= \frac{4x_2^2 - x_2 x_3}{12,566(x_2 x_3^3 - x_3^4)} + \frac{2.46}{12,566 x_3^2} - 1 \leq 0 \\
w_3(\boldsymbol{x}) &= 1 - \frac{140.54 x_3}{x_2^2 x_1} \leq 0 \\
w_4(\boldsymbol{x}) &= \frac{x_2 + x_3}{1.5} - 1 \leq 0 \\
& 2 \leq x_1 \leq 15, 0.25 \leq x_2 \leq 1.30, 0.05 \leq x_3 \leq 0.20.
\end{aligned} \tag{18}
$$

This problem has a very small feasible area: sampling 100,000 points using an LHS with midpoints and evaluating these with the true constraint functions $w_{h'}(\boldsymbol{x})$ ($h' = 1, ..., 4$) in (18), we estimate it to be approx. 9.7% of the total experimental area.

Most publications using this test problem only report the final goal value; Kazemzadeh-Parsi (2014) (Table 6) is one of the few that also gives the input location of the optimum ($\widehat{\boldsymbol{x}}_{\mathrm{o}} \approx (11.25950, 0.35770, 0.05173)$), so we take this as the reference optimum. Note that $\widehat{\boldsymbol{x}}_{\mathrm{o}}$ satisfies the input constraints, with $\widehat{x}_{\mathrm{o};3} = 0.05173$ close to its lower bound 0.05. This solution yields $w_0 \approx 0.01269$; $w_1 \approx -0.0012$, $w_2 \approx 0.0000$, $w_3 \approx -4.0464$ and $w_4 \approx -0.7270$: all output constraints are thus satisfied, with $w_2(\widehat{\boldsymbol{x}}_{\mathrm{o}})$ being binding.

Given $k = 3$, we start with an initial LHS design of $n = (k+1)(k+2)/2 = 10$ points; again, we perform 50 macroreplications, each time starting from an independent LHS design. Fig. 6 illustrates

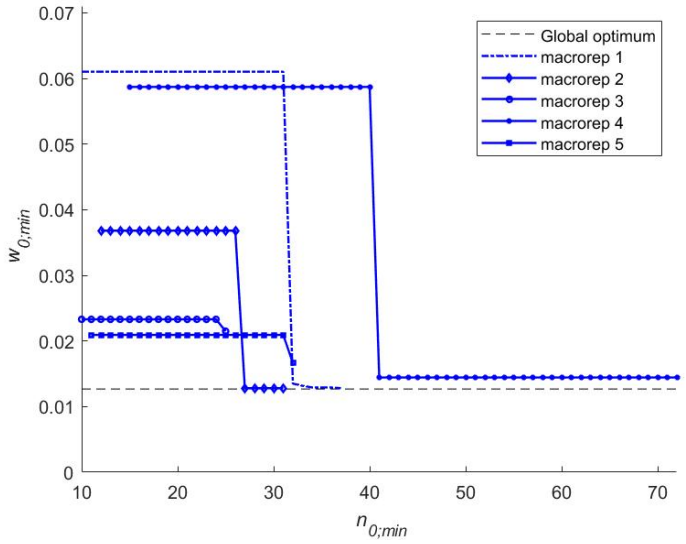


Figure 6 Best found goal value in terms of the number of expensive evaluations, for the first five macroreplications of the spring problem.

that many infill points sampled at the beginning of the algorithm turn out to be infeasible (this is shown only for the first 5 macroreplications; the first parts of the curves are horizontal). Yet, as KT-EGO learns the I/O behavior of the model, the OK metamodels improve, resulting in feasible infill points (with improving goal values) in the later iterations.

The boxplots in Figure 7 show that, after 50 macroreplications, the mean best found goal value equals 0.0154 (this mean is biased upwards due to 1 outlier; the median best goal value found equals 0.0134). The average number of expensive evaluations required amounts to 40.36 (again due to the single outlier), with a median of 32 expensive evaluations (including 10 initial design evaluations). Comparing these results with the results of the EKCO algorithm by (Tao et al. 2020), which show an average best found goal value approximately equal to 0.01267 (with a median of 0.01267), and an average number of expensive evaluations of 92.5 (median equal to 97), we can conclude that our algorithm again compares favorably on speed, while the median best found goal value remains quite good; occasionally, outliers may again occur. Recall also that the algorithm by Tao et al. (2020) reports the Kriging output of the *estimated Kriging optimum* (which we refer to as $\mathbf{x_{est}}$, Step 9 of our algorithm) as the final optimum, which differs from our approach.

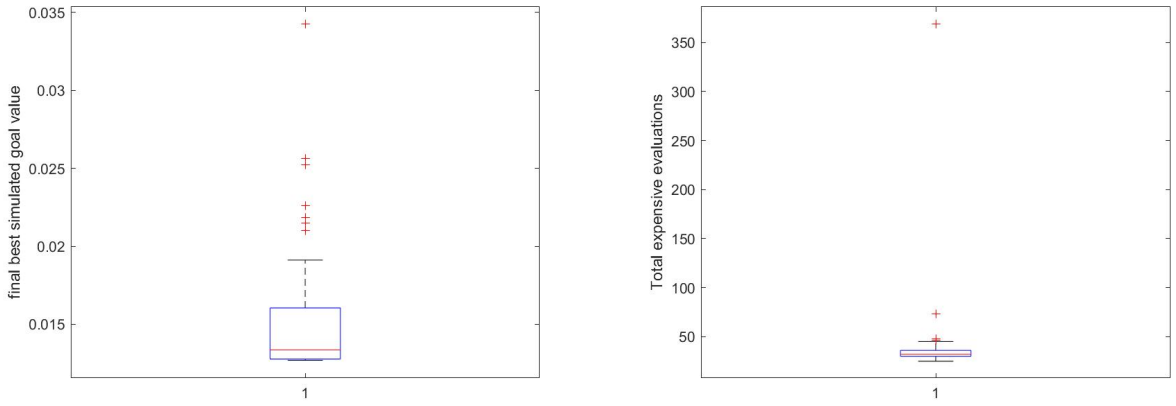


Figure 7 Boxplots of the final best goal value found (left pane) and the number of expensive evaluations required (right pane), over 50 macroreplications, for the spring problem.

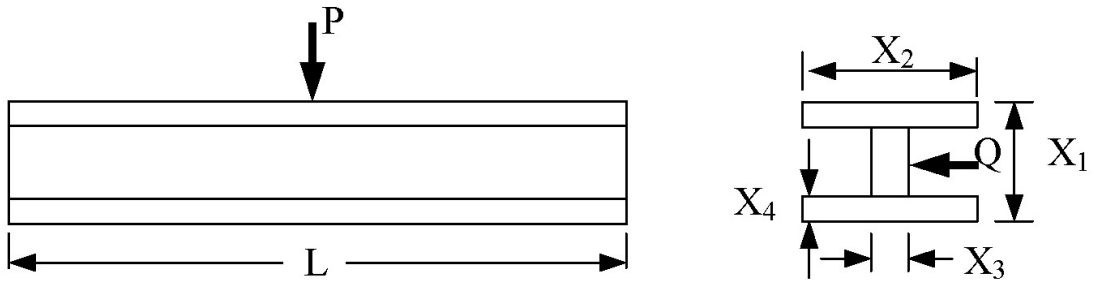


Figure 8 I-beam design problem(Wang 2003).

6.3. I-beam design problem

Wang (2003) presents the engineering model of the *I-beam design* (IBD) problem that we reproduce in Fig. 8.

The objective is to minimize the vertical deflection of an I-beam, subject to cross-section area and stress constraints under given loads (Tao et al. (2020), Supplement, Problem (18)):

$$\min_{\boldsymbol{x}} w_0(\boldsymbol{x}) = \frac{5000}{(1/12)x_3(x_1-2x_4)^3+(1/6)x_2x_4^3+2x_2x_4[(x_1-x_4)/2]^2}$$

$$s.t. w_1(\boldsymbol{x}) = 2x_2x_4+x_3(x_1-2x_4)-300 \leq 0$$

$$w_2(\boldsymbol{x}) = \frac{180,000x_1}{x_3(x_1-2x_4)^3+2x_2x_4(4x_4^2+3x_1(x_1-2x_4)}+$$

$$+\frac{15,000x_2}{(x_1-2x_4)x_3^3+2x_4x_2^3}-6\leq 0$$

$$10\leq x_1\leq 80, 10\leq x_2\leq 50, 0.9\leq x_3\leq 5, 0.9\leq x_4\leq 5. \quad (19)$$

Using an LHS with $100,000$ points, we estimate that the IBD example has a feasible area that is less than 1% of the experimental area. Consequently, this is a challenging test problem for black-box optimization algorithms.

KT-EGO uses an initial design of $n=(k+1)(k+2)/2=15$ points to analyze this problem. Figure 9 illustrates the best found goal value in terms of the number of iterations performed, for the first 5 macroreplications: in spite of the extremely small feasible region (in fact, none of these macroreplications contained a single feasible point in their input design), our algorithm tends to find a feasible solution within a very small number of iterations, and succeeds to improve on this solution relatively fast. Figure 10 shows the boxplots for the best found goal value and the number of expensive evaluations required, after 50 macroreplications. We obtain a mean goal value of 0.0132, with a median of 0.0131. The mean number of expensive evaluations equals 52.02, with a median of 43.5.The EKCO algorithm by Tao et al. (2020), Tables 3 and 7, gives a mean equal to 0.01307 (and the same median) for the best goal value found, and a mean number of expensive evaluations equal to 49 (with a median of 50).

7. Conclusions

We derived a novel method for solving constrained optimization problems in black-box deterministic simulation. To the best of our knowledge, this algorithm is the first to combine the well-known Expected Improvement criterion (which balances local exploitation and global exploration) with Karush-Kuhn-Tucker conditions in a single infill criterion. To quantify how well these conditions hold, our approach uses Ordinary Kriging and least squares regression. Pattern search was used in the numerical experiments, to optimize this criterion. The results have shown that we may expect

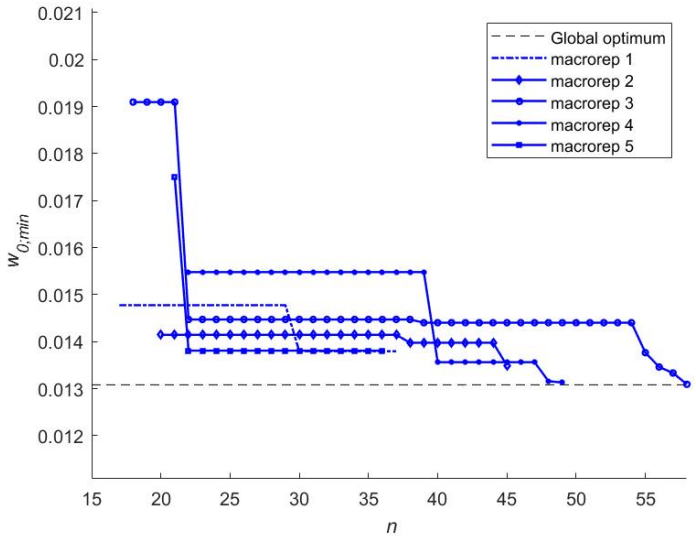


Figure 9 Best found goal value ($w_{0;min}$) in terms of the number of expensive evaluations, for the Ibeam problem.

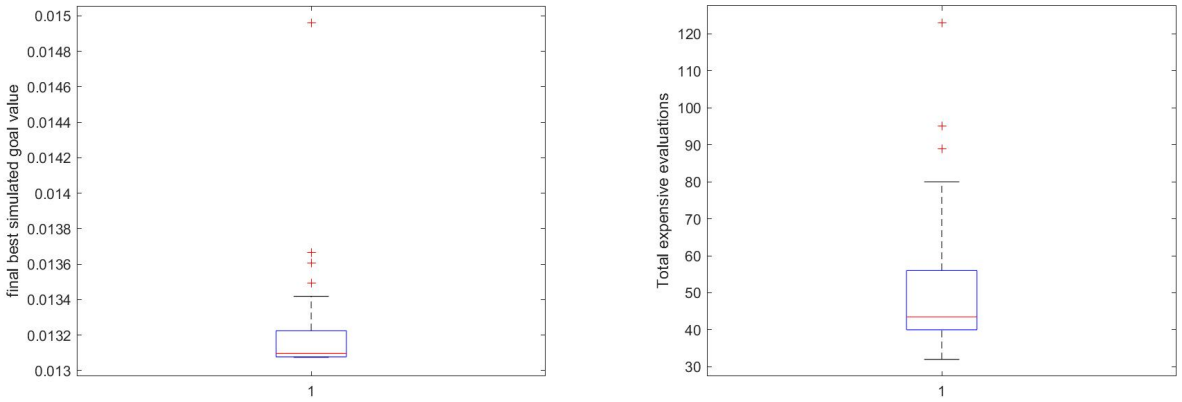


Figure 10 Boxplots of the final best goal value found (left pane) and the number of expensive evaluations required (right pane), over 50 macroreplications, for the Ibeam problem.

our algorithm to give feasible solutions that are "close" to the global optimum,requiring fewer (expensive) simulations than competing algorithms do (we compared KT-EGO with three recent alternatives for the toy problem, and with results of the EKCO algorithm by Tao et al. (2020) for the spring and the I-beam problem). The algorithm sequentially samples the local area that is estimated to be near the binding constraints: while the expected improvement factor tends to

select the next infill point in the infeasible area, the KKT factor successfully corrects this tendency. The only downside observed is that, occasionally, our algorithm stops too early.

Acknowledgments

We thank the following colleagues for accepting our invitation to apply their EGO variants to the toy example that was originally defined in Gramacy et al. (2016), and share their results with us: 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 Ebru Angün (Galatasaray University, Turkey) and Dick den Hertog (University of Amsterdam) for clarifying the KT conditions. This research was supported by the Flanders Artificial Intelligence Research Program (FLAIR).

References

Bagheri S, Konen W, Allmendinger R, Branke J, Deb K, Fieldsend J, Quagliarella D, Sindhya K (2017) Constraint handling in efficient global optimization. *Proceedings of GECCO '17* (Berlin).

Boyd S, Vandenberghe L (2009) *Convex optimization* (Cambridge University Press, United Kingdom).

Calma A, Ho W, Shao L, Li H (2021) Operations research: topics, impact, and trends from 1952–2019. *Operations Research* 69:1487–1508.

Carpio R, Giordano RC, Secchi AR (2018) Enhanced surrogate assisted framework for constrained global optimization of expensive black-box functions. *Computers & Chemical Engineering*, 118: 91–102.

Chen J, Kang L, Lin G (2021) Gaussian process assisted active learning of physical laws. *Technometrics*, 63(3): 329–342.

Gramacy RB, Gray GA, Le Digabel S, Lee HKH, Ranjan P, Wells G, Wild SM (2016) Modeling an augmented Lagrangian for blackbox constrained optimization, *Technometrics*, 58(1):1–11.

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

Kazemzadeh-Parsi MJ (2014), A modified firefly algorithm for engineering design optimization problems. *Iranian Journal of Science and Technology: Transactions of Mechanical Engineering*, 38:403–421.

Kleijnen JPC, Mehdad E (2014), Multivariate versus univariate Kriging metamodels for multi-response simulation models. *European Journal of Operational Research*, 236(2):573–582.

Kleijnen JPC (2015) *Design and analysis of simulation experiments.* (Springer)

Kleijnen JPC (2021), Kriging: methods and applications. *Model Order Reduction; Volume 1: System- and Data-Driven Methods and Algorithms*, edited by P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L.M. Silveira, De Gruyter (Berlin).

Kleijnen JPC, van Beers WCM (2021) Statistical Tests for Cross-Validation of Kriging Models. *INFORMS Journal on Computing* 34(1):607–621.

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

Loeppky JL, Sacks J, Welch W (2009) Choosing the sample size of a computer experiment: a practical guide. *Technometrics*, 51(4):366–376.

Picheny V, Gramacy RB, Wild S, Le Digabel S (2016) Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. *Advances in Neural Information Processing Systems Conference* (NIPS 29)

Pourmohamad T, Lee HKH(2020), The statistical filter approach to constrained optimization. *Technometrics*, 62(3):303–312.

Pourmohamad T, Lee HKH(2022), Bayesian optimization via barrier functions.*Journal of Computational and Graphical Statistics*, 31(1):74–83.

Rasmussen CE, Williams CKI (2006) *Gaussian processes for machine learning* (MIT Press).

Sadoughi M, Li M, Hu C (2018) Multivariate system reliability analysis considering highly nonlinear and dependent safety events. *Reliability Engineering & System Safety*,180:189–200.

Tao I, Zhao G, Ren S (2020) An efficient Kriging-based constrained optimization algorithm by global and local sampling in feasible region. *Journal of Mechanical Design*, 142(5):051401-1– 051401-15.

Tran A, Wildey T, Mccann S (2019) sBF-BO-2CoGP: A sequential bi-fidelity constrained Bayesian optimization for design applications. *Proceedings of the ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 1: 39th Computers and Information in Engineering Conference.* Anaheim, California, USA. August 18–21, 2019.

Wang GG (2003) Adaptive response surface method using inherited Latin hypercube design. *Transactions of the ASME, Journal of Mechanical Design*, 125:210–220.