Mathematical Programming Computation (2021) 13:553–581
https://doi.org/10.1007/s12532-021-00204-y

FULL LENGTH PAPER

Deterministic global optimization with Gaussian processes embedded

Artur M. Schweidtmann[1] · Dominik Bongartz[1] · Daniel Grothe[1] · Tim Kerkenhoff[1] · Xiaopeng Lin[1] · Jaromił Najman[1] · Alexander Mitsos[1,2,3]

Received: 20 May 2020 / Accepted: 28 May 2021 / Published online: 25 June 2021

Abstract

Gaussian processes (Kriging) are interpolating data-driven models that are frequently applied in various disciplines. Often, Gaussian processes are trained on datasets and are subsequently embedded as surrogate models in optimization problems. These optimization problems are nonconvex and global optimization is desired. However, previous literature observed computational burdens limiting deterministic global optimization to Gaussian processes trained on few data points. We propose a reduced-space formulation for deterministic global optimization with trained Gaussian processes embedded. For optimization, the branch-and-bound solver branches only on the free variables and McCormick relaxations are propagated through explicit Gaussian process models. The approach also leads to significantly smaller and computationally cheaper subproblems for lower and upper bounding. To further accelerate convergence, we derive envelopes of common covariance functions for GPs and tight relaxations of acquisition functions used in Bayesian optimization including expected improvement, probability of improvement, and lower confidence bound. In total, we reduce computational time by orders of magnitude compared to state-of-the-art methods, thus overcoming previous computational burdens. We demonstrate the performance and scaling of the proposed method and apply it to Bayesian optimization with global optimization of the acquisition function and chance-constrained programming. The Gaussian process models, acquisition functions, and training scripts are available open-source within the "MeLOn—**M**achin**e** **L**earning Models for **O**ptimizatio**n**" toolbox (https://git.rwth-aachen.de/avt.svt/public/MeLOn).

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—Cluster of Excellence 2186 "The Fuel Science Center" and the project "Improved McCormick Relaxations for the efficient Global Optimization in the Space of Degrees of Freedom" MI 1851/4-1.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12532-021-00204-y.

Extended author information available on the last page of the article

Keywords Kriging · Machine learning · Bayesian optimization · Acquisition function · Chance-constrained programming · Reduced-space · Expected improvement

Mathematics Subject Classification 90C26 · 90C30 · 90C90 · 68T01 · 60-04

1 Introduction

A Gaussian process (GP) is a stochastic process where any finite collection of random variables has a multivariate Gaussian distribution; they can be understood as an infinite-dimensional generalization of multivariate Gaussian distributions [66]. The predictions of GPs are Gaussian distributions that provide not only an estimate but also a variance. GPs originate from geostatistics [46] and gained popularity for the design and analysis of computer experiments (DACE) since 1989 [68]. Furthermore, GPs are commonly applied as interpolating surrogate models across various disciplines including biotechnology [13,25,29,54,82], chemical engineering [14,22–24,27,34,52], chemistry [1,69], and deep-learning [74]. Note that GP regression is also often referred to as Kriging. In many applications, GPs are trained on a data set and are subsequently embedded in an optimization problem, e.g., to identify an optimal operating point of a process. Moreover, many derivative-free solvers for expensive-to-evaluate black-box functions actually train GPs and optimize their predictions (e.g., Bayesian optimization algorithms [12,40,72,82] and other adaptive sampling approaches [9,10,20,22–24,27]). In Bayesian optimization, the optimum of an acquisition function determines the next sampling point [72]. The vast majority of these optimizations have been performed by local solution approaches [44,88] and a few by stochastic global optimization methods [12]. Our contribution focuses on the deterministic global solution of optimization problems with trained GPs embedded and on applications in process systems engineering.

GPs are commonly used to learn the input-output behavior of unit operations [14,15,42,43,48,63,64], complete flowsheets [33], or thermodynamic property relations from data [51]. Subsequently, the trained GPs are often combined with nonlinear mechanistic process models leading to hybrid mechanistic and data-driven models [31,41,60,83] which are optimized. Many of the previous works on optimization with GPs embedded rely on local optimization techniques. Caballero and Grossmann, for instance, train GPs on data obtained from a rigorous divided wall column simulation. Then, they iteratively optimize the operation of the column (modeled by GPs) using SNOPT [30], sample new data at the solution point, and update the GPs [14,15]. Later, Caballero and co-workers extend this work to distillation sequence superstructure problems [63,64]. In [63], the authors solve the resulting mixed-integer nonlinear programs (MINLPs) using a local solver in GAMS. Therein, the GP estimate is computed via an external function in Matlab which leads to a reduced optimization problem size visible to the local solver in GAMS. However, all these local methods have the drawback that they can lead to suboptimal solutions, because the resulting optimization problems are nonconvex. This nonconvexity is induced by the covariance functions of the GPs as well as often the mechanistic part of the hybrid models.

Deterministic global optimization can guarantee to identify globally optimal solutions within finite time to a given nonzero tolerance [36]. In a few previous studies, deterministic global optimization with GPs embedded was done using general-purpose global solvers. For instance, in the black-box optimization algorithms ALAMO [20] and ARGONAUT [9], GPs are included as surrogate models and are optimized using BARON [79] and ANTIGONE [57], respectively. However, computational burdens were observed that limit applicability, e.g., in terms of the number of training points. Cozad et al. [20] state that GPs are accurate but "difficult to solve using provable derivative-based optimization software". Similarly, Boukouvala and Floudas [9] state that the computational cost becomes a limiting factor because the number of nonlinear terms of GPs equals the product of the number of interpolated points (N) and the dimensionality (D) of the input domain. More recently, Keßler et al. [42,43] optimized the design of nonideal distillation columns by a trust-region approach with GPs embedded. Therein, optimization problems with GPs embedded are solved globally using BARON [79] within relatively long CPU times (10^2–10^5 CPU seconds on a personal computer).

As mentioned earlier, Quirante et al. [63] call an external Matlab function to compute GP estimates with a local solver in GAMS. As an alternative approach, they also solve the problem globally using BARON in GAMS by providing the full set of GP equations as equality constraints. This leads to additional intermediate optimization variables besides the degrees of freedom of the problem. Similar to other studies, they observe that their formulation is only practical for a small number of GP surrogates and training data points to avoid large numbers of variables and constraints [63]. We refer to the problem formulation where the GP is described by equality constraints and additional optimization variables as a full-space (FS) formulation. It is commonly used in modeling environments, e.g., GAMS, that interface with state-of-the-art global solvers such as ANTIGONE [57], BARON [79], and SCIP [50].

An alternative to the FS is a reduced-space (RS) formulation where some optimization variables are eliminated using explicit constraints. This reduced problem size leads to a lower number of variables for branching as well as potentially smaller subproblems. The former has some similarity to selective branching [28] (c.f. discussion in [4]). The exact size of the subproblems for lower bounding and bound tightening depends on the method for constructing relaxations. In particular, when constructing relaxations in the RS using McCormick [53], alphaBB [2] or natural interval extensions, the resulting lower bounding problems are much smaller compared to the auxiliary variable method (AVM) [73,80]. Therefore, any global solver can in principle handle RS but some methods for constructing relaxations appear more promising to benefit from the RS [4]. We have recently released the open-source global solver MAiNGO [7] which uses the MC++ library [16] for automatic propagation of McCormick relaxations through computer code [58]. We have shown that the RS formulation can be advantageous for flowsheet optimization problems [5,6] and problems with artificial neural networks embedded [37,65,70]. In the context of Bayesian optimization, Jones et at. [40] develop valid overestimators of the expected improvement (EI) acquisition function in the RS. However, their relaxations rely on interval extensions and optimization-based relaxations limited to a specific covariance function; they do not derive envelopes. Furthermore, they do not provide convex underestimators which are in general necessary to embed GPs in optimization problems.

The main contribution of this work is the efficient deterministic global optimization of optimization problems with GPs embedded. We develop a RS formulation for optimization problems with GPs embedded. The performance of the proposed method is analyzed in an extensive computational study by solving about 90,000 optimization problems. The proposed RS outperforms a FS formulation for problems with GPs embedded by speedup factors of several magnitudes. Moreover, this speedup increases with the number of training points. To further accelerate convergence, we derive and implement envelopes of covariance functions for GPs and tight relaxations of acquisition functions, which are commonly used in Bayesian optimization. Finally, we solve a chance-constrained optimization problem with GPs embedded and we perform global optimization of an acquisition function. The GP training methods and models are provided as an open-source toolbox called "MeLOn—**M**achin**e** **L**earning Models for **O**ptimizatio**n**" under the Eclipse public license [71]. The resulting optimization problems are solved using our global solver MAiNGO [7]. Note that the MeLOn toolbox is also automatically included as a submodule in our new MAiNGO release.

2 Optimization problem formulations

In the simplest case, which is common in the literature, the (scaled) inputs of a GP are the free variables of the optimization problem with $\boldsymbol{x} \in \tilde{X} = [\boldsymbol{x}^L, \boldsymbol{x}^U]$. For given $\boldsymbol{x}$, the dependent (or intermediate) variables $\boldsymbol{z}$ can be computed by the solution of $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{0}, \quad \boldsymbol{h} : \tilde{X} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$. In the case of GP models, we aim to include the estimate ($m_{\mathcal{D}}$) and variance ($k_{\mathcal{D}}$) in the optimization. As will be shown in Sect. 3, we can solve explicitly for $m_{\mathcal{D}}$ and $k_{\mathcal{D}}$ [c.f. Eqs. (1) and (2)].

The realization of the objective function f depends on the application. In many applications, it depends on the estimate of the GP, i.e., $f(m_{\mathcal{D}})$ (c.f. Sect. 6.1). In Bayesian optimization, the objective function is called the acquisition function and usually depends on the estimate and variance of the GP, i.e., $f(m_{\mathcal{D}}, k_{\mathcal{D}})$ (c.f. Sect. 6.3). Finally, additional constraints might depend on the inputs of the GP, its estimate, and variance, i.e., $\boldsymbol{g}(\boldsymbol{x}, m_{\mathcal{D}}, k_{\mathcal{D}}) \leq \boldsymbol{0}$. In more complex cases, multiple GPs can be combined in one optimization problem (c.f. Sect. 6.2).

In the following, we describe two optimization problem formulations for problems with trained GPs embedded: the commonly used FS formulation in Sect. 2.1 and the RS formulation in Sect. 2.2. Both problem formulations are exact reformulations in the sense of Liberti et al. [47], meaning that they have the same local and global optima. The equivalence is shown in "Appendix A" of [4]. However, the formulation significantly affects problem size and performance of global optimization solvers.

2.1 Full-space formulation

In the FS formulation, the nonlinear equations $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{0}$ are provided as equality constraints and the intermediate dependent variables $\boldsymbol{z} \in Z$ are optimization variables. A general FS problem formulation is:

$$\begin{aligned} \min_{\boldsymbol{x}\in\tilde{X},\boldsymbol{z}\in Z} \quad & f(\boldsymbol{x},\boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{0}, \qquad \boldsymbol{g}(\boldsymbol{x},\boldsymbol{z}) \leq \boldsymbol{0} \end{aligned} \tag{FS}$$

In general, there exist multiple valid FS formulations for optimization problems. In Sect. 3 of the electronic supplementary information (ESI), we provide a representative FS formulation for the case where the estimate of a GP is minimized. This is also the FS formulation that we use in our numerical examples (c.f., Sect. 6.1).

2.2 Reduced-space formulation

In the RS formulation, the equality constraints are solved for the intermediate variables and substituted in the optimization problem (c.f. [5]). A general RS problem formulation in the context of optimization with a GP embedded is:

$$\begin{aligned} \min_{\boldsymbol{x}\in\tilde{X}} \quad & f(m_{\mathcal{D}}(\boldsymbol{x}), k_{\mathcal{D}}(\boldsymbol{x})) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}, m_{\mathcal{D}}(\boldsymbol{x}), k_{\mathcal{D}}(\boldsymbol{x})) \leq \boldsymbol{0} \end{aligned} \tag{RS}$$

Herein, the Branch-and-Bound (B&B) solver operates only on the free variables $\boldsymbol{x}$ and no equality constraints are visible to the solver. In GPs, the estimate and variance are explicit functions of the input [Eqs. (1) and (2)]. Thus, we can directly formulate a RS formulation. The RS formulation effectively combines those equations and hides them from the B&B algorithm. This results in a total number of D optimization variables, zero equality constraints, and no additional optimization variables $\boldsymbol{z}$. Thus, the RS formulation requires only bounds on $\boldsymbol{x}$.

Note that the direct substitution of all equality constraints is not always possible when multiple GPs are combined with mechanistic models, e.g., in the presence of recycle streams. Here, a small number of additional optimization variables and corresponding equality constraints can remain in the RS formulation [5]. As an alternative, relaxations for implicit functions can also be derived [76,86]. Moreover, we have previously observed that a hybrid between RS and FS formulation can be more efficiently solvable for some optimization problems [6]. In this work, we compare the RS and the FS formulation and do not consider any hybrid problem formulations.

3 Gaussian processes

In this section, GPs are briefly introduced (c.f. [66]). We first describe the GP prior distribution, i.e., the probability distribution before any data is taken into account. Then, we describe the posterior distribution, which results from conditioning the prior on training data. Finally, we describe how hyperparameters of the GP can be adapted to data by a maximum a posteriori (MAP) estimate.

3.1 Prior

A GP prior is fully described by its mean function $m(\mathbf{x})$ and positive semi-definite covariance function $k(\mathbf{x}, \boldsymbol{x}')$ (also known as kernel function). We consider a noisy observation y from a function $\tilde{f}(\mathbf{x})$ with $y(\mathbf{x}) := \tilde{f}(\mathbf{x}) + \varepsilon_{\text{noise}}$, whereby the output noise $\varepsilon_{\text{noise}}$ is independent and identically distributed (i.i.d.) with $\varepsilon_{\text{noise}} \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. We say y is distributed as a GP, i.e., $y \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ with

$$\begin{aligned} m(\mathbf{x}) &:= \mathbb{E}\big[\tilde{f}(\mathbf{x})\big], \\ k(\mathbf{x}, \boldsymbol{x}') &:= \mathbb{E}\big[\ (y(\mathbf{x}) - m(\mathbf{x}))\ (y(\mathbf{x}') - m(\mathbf{x}))^{\mathrm{T}}\big]. \end{aligned}$$

Without loss of generality, we assume that the prior mean function is $m(\mathbf{x}) = 0$. This implies that we train the GP on scaled data such that the mean of the training outputs is zero. A common class of covariance functions is the Matérn class.

$$k_{\text{Matérn}}(\boldsymbol{x}, \boldsymbol{x}') := \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} r\right)^{\nu} K_\nu \left(\sqrt{2\nu} r\right),$$

where σ_f^2 is the output variance, $r := \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} \boldsymbol{\Lambda}\ (\boldsymbol{x} - \boldsymbol{x}')}$ is a weighted Euclidean distance, $\boldsymbol{\Lambda} := \text{diag}(\lambda_1^2, \cdots, \lambda_i^2, \cdots \lambda_{n_x}^2)$ is a length-scale matrix with $\lambda_i \in \mathbb{R}$, $\Gamma(\cdot)$ is the gamma function, and $K_\nu(\cdot)$ is the modified Bessel function of the second kind. The smoothness of Matérn covariance functions can be adjusted by the positive parameter ν. When ν is a half-integer value, the Matérn covariance function becomes a product of a polynomial and an exponential [66]. Common values for ν are $1/2,\ 3/2,\ 5/2$, and ∞, i.e., the most widely-used squared exponential covariance function, $k'_{SE}(r) := \exp\left(-\frac{1}{2}\, r^2\right)$. We derive envelopes of these covariance functions in Sect. 4.1 and implement them within MeLOn [71]. Also, a noise term, $\sigma_{\text{n}}^2 \cdot \delta(\boldsymbol{x}, \boldsymbol{x}')$, can be added to any covariance function where σ_{n}^2 is the noise variance and $\delta(\boldsymbol{x}, \boldsymbol{x}')$ is the Kronecker delta function. The hyperparameters of the covariance function are adjusted during training and are jointly noted as $\boldsymbol{\theta} = [\lambda_1, ..., \lambda_d, \sigma_f, \sigma_{\text{n}}]$. Herein, a log-transformation is common to prevent negative values during training.

3.2 Posterior

The GP posterior is obtained by conditioning the prior on observations. We consider a set of N training inputs $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}_1^{(\boldsymbol{\mathcal{D}})}, ..., \boldsymbol{x}_N^{(\boldsymbol{\mathcal{D}})}\}$ where $\boldsymbol{x}_i^{(\boldsymbol{\mathcal{D}})} = [x_{i,1}^{(\boldsymbol{\mathcal{D}})}, ..., x_{i,D}^{(\boldsymbol{\mathcal{D}})}]^T$ is a D-dimensional vector. Note that we use the superscript $(\boldsymbol{\mathcal{D}})$ to denote the training data. The corresponding set of scalar observations is given by $\boldsymbol{\mathcal{Y}} = \{y_1^{(\boldsymbol{\mathcal{D}})}, ..., y_N^{(\boldsymbol{\mathcal{D}})}\}$. Furthermore, we define the vector of scalar observations $\boldsymbol{y} = [y_1^{(\boldsymbol{\mathcal{D}})}, ..., y_N^{(\boldsymbol{\mathcal{D}})}]^T \in \mathbb{R}^N$. The posterior GP is obtained by Bayes' theorem:

$$\tilde{f}(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}') | \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}) = \mathcal{N}\left(m_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x}), k_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x}, \boldsymbol{x}')\right)$$

with

$$m_{\mathcal{D}}(\boldsymbol{x}) = \boldsymbol{K}_{\boldsymbol{x},\mathcal{X}} \left(\boldsymbol{K}_{\mathcal{X},\mathcal{X}}\right)^{-1} \boldsymbol{y}, \tag{1}$$

$$k_{\mathcal{D}}(\boldsymbol{x}) = K_{\boldsymbol{x},\boldsymbol{x}} - \boldsymbol{K}_{\boldsymbol{x},\mathcal{X}} \left(\boldsymbol{K}_{\mathcal{X},\mathcal{X}}\right)^{-1} \boldsymbol{K}_{\mathcal{X},\boldsymbol{x}}, \tag{2}$$

where the covariance matrix of the training data is given by $\boldsymbol{K}_{\mathcal{X},\mathcal{X}} := \left[k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right] \in \mathbb{R}^{N\times N}$, the covariance vector between the candidate point $\boldsymbol{x}$ and the training data is given by $\boldsymbol{K}_{\boldsymbol{x},\mathcal{X}} := \left[k(\boldsymbol{x}, \boldsymbol{x}_1^{(\mathcal{D})}), ..., k(\boldsymbol{x}, \boldsymbol{x}_N^{(\mathcal{D})})\right] \in \mathbb{R}^{1\times N}$, $\boldsymbol{K}_{\mathcal{X},\boldsymbol{x}} = \boldsymbol{K}_{\boldsymbol{x},\mathcal{X}}{}^T$, and $K_{\boldsymbol{x},\boldsymbol{x}} := k(\boldsymbol{x}, \boldsymbol{x})$. Equations (1) and (2) describe essentially the predictions of a GP and are implemented within MeLOn.

3.3 Maximum a posteriori

In order to find appropriate hyperparameters $\boldsymbol{\theta}$ for a given problem, we use a MAP estimate which is known to be advantageous compared to the maximum likelihood estimation (MLE) on small data sets [77]. Using the MAP estimate, the hyperparameters are identified by maximizing the probability that the GP fits the training data, i.e., $\boldsymbol{\theta}_{\text{opt}} := \text{argmax}_{\boldsymbol{\theta}}\, \mathcal{P}\left(\boldsymbol{\theta} | \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}\right)$. Analytical expressions for $\mathcal{P}\left(\boldsymbol{\theta} | \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}\right)$ and its derivatives w.r.t. the hyperparameters can be found in the literature [66]. We provide a Matlab training script in MeLOn that is based on our previous work [12]. Therein, we assume an independent Gaussian distribution as a prior distribution on the log-transformed hyperparameters, i.e., $\theta_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right)$. The implementation of the training is efficient through the pre-computation of squared distances, the Cholesky decomposition for computing the inverse of the covariance matrix, and a two-step training approach that searches first globally and then locally [12].

4 Convex and concave relaxations

The construction of relaxations, i.e., convex function underestimators (F^{cv}) and concave function overestimators (F^{cc}), is essential for B&B algorithms. In our open-source solver MAiNGO, we use the (multivariate) McCormick method [53,81] to propagate relaxations and their subgradients [58] through explicit functions using the MC++ library [16]. However, the McCormick method often does not provide the tightest possible relaxations, i.e., the envelopes. In this section, we derive tight relaxations or envelopes of functions that are relevant for GPs and Bayesian optimization. The functions and their relaxations are implemented in MC++. When using these intrinsic functions and their relaxations in MAiNGO, the (multivariate) McCormick method is only used for the remaining parts of the model. Note that the derived relaxations are used within MAiNGO while BARON does not allow for implementation of custom relaxations or piecewise defined functions.

4.1 Covariance functions

The covariance function is a key element of GPs. When embedding trained GPs into optimization problems, the covariance function occurs N times because it is used in the covariance vector between the candidate point $\boldsymbol{x}$ and the training data, i.e., $\boldsymbol{K}_{\boldsymbol{x},\mathcal{X}} = \left[k(\boldsymbol{x}, \boldsymbol{x}_1^{(\mathcal{D})}), ..., k(\boldsymbol{x}, \boldsymbol{x}_N^{(\mathcal{D})})\right] \in \mathbb{R}^{1\times N}$. Note that the covariance matrix $\boldsymbol{K}_{\mathcal{X},\mathcal{X}}$ depends only on training data and is thus a parameter during the optimization. Thus, tight relaxations of the covariance functions are highly desirable. In this subsection, we derive envelopes for common Matérn covariance functions. We consider univariate covariance functions, i.e., $k_\nu : \mathbb{R} \to \mathbb{R}$, with input $d = (\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} \boldsymbol{\Lambda}\ (\boldsymbol{x} - \boldsymbol{x}') \geq 0$. This is possible because we consider stationary covariance functions that are invariant to translations in the input space. Common Matérn covariance functions use $\nu = 1/2,\ 3/2,\ 5/2$ and ∞ and are given by:

$$k_{\nu=1/2}(d) := \exp\left(-\sqrt{d}\right), \qquad k_{\nu=3/2}(d) := \left(1 + \sqrt{3}\,\sqrt{d}\right) \cdot \exp\left(-\sqrt{3}\,\sqrt{d}\right)$$
$$k_{\nu=5/2}(d) := \left(1 + \sqrt{5}\,\sqrt{d} + \frac{5}{3}\,d\right) \cdot \exp\left(-\sqrt{5}\sqrt{d}\right), \qquad k_{SE}(d) := \exp\left(-\frac{1}{2}\,d\right),$$

where k_{SE} is the squared exponential covariance function with $\nu \to \infty$. We find that these four covariance functions are convex because their Hessian is positive semidefinite. Thus, the convex envelope is given by $F^{cv}(d) = k(d)$ and the concave envelope by the secant $F^{cc}(d) = \mathrm{sct}(d)$ where $\mathrm{sct}(d) = \frac{k(d^U)-k(d^L)}{d^U-d^L} d + \frac{d^U k(d^L)-d^L k(d^U)}{d^U-d^L}$ on a given interval $[d^L, d^U]$. As the McCormick composition and product theorems provide weak relaxations of $k_{\nu=3/2}$ and $k_{\nu=5/2}$ (c.f. ESI Sect. 1), we implement these functions and their envelopes in our library of intrinsic functions in MC++. Furthermore, natural interval extensions are not exact for $k_{\nu=3/2}$ and $k_{\nu=5/2}$. Thus, we also provide exact interval bounds based on the monotonicity.

It should be noted that covariance functions are commonly given as a function of the weighted Euclidean distance $r = \sqrt{d}$. However, we chose to use d instead for three main reasons: (1) $\boldsymbol{x}$ is usually a free variable of the optimization problem. Thus, the computation of r would lead to potentially weaker relaxations for $k_{\nu=5/2}$ and k_{SE}. (2) The derivative of $k_{\nu=3/2}(\cdot)$, $k_{\nu=5/2}(\cdot)$, and $k_{SE}(\cdot)$ is defined at $d = 0$ while the derivative of the square root function is not. (3) The covariance functions $\hat{k}_{\nu=3/2} : r \mapsto k_{\nu=3/2}(r^2)$, $\hat{k}_{\nu=5/2} : r \mapsto k_{\nu=5/2}(r^2)$, and $\hat{k}_{SE} : r \mapsto k_{SE}(r^2)$ are nonconvex in r, so deriving the envelopes would be nontrivial.

Finally, it can be noted that we did not derive envelopes of $k_{\text{Matérn}}(\boldsymbol{x}, \boldsymbol{x}')$, because the variable input dimensions pose difficulties in implementation and the multidimensionality is a challenge for the derivation of envelopes. Nevertheless, the McCormick composition theorem applied to $k_\nu(d(\boldsymbol{x}, \boldsymbol{x}'))$ yields relaxations that are exact at the minimum of $k_{\text{Matérn}}$ because the natural interval extensions of the weighted squared distance d are exact (c.f. [62]). This means that the relaxations are exact in Hausdorff metric.

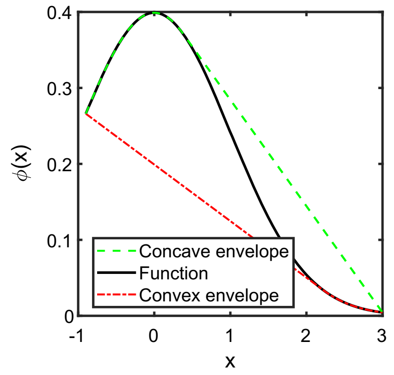


Fig. 1 Illustration of the envelope of the Gaussian PDF

4.2 Gaussian probability density function

The PDF is used to compute the EI acquisition function and is given by $\phi : \mathbb{R} \rightarrow \mathbb{R}$ with

$$\phi(x) := \frac{1}{\sqrt{2\pi}} \cdot \exp\left(\frac{-x^2}{2}\right) \tag{3}$$

The Gaussian probability density function (PDF) is a nonconvex function for which the McCormick composition rule does not provide its envelopes. For one-dimensional functions, McCormick [53] also provides a method to construct envelopes. We construct the envelopes of PDF using this method and implement them in our library of intrinsic functions. The envelope of the PDF is illustrated in Fig. 1 and derived in "Appendix A.1".

4.3 Gaussian cumulative distribution function

The Gaussian cumulative distribution function (CDF) is given by $\Phi : \mathbb{R} \rightarrow \mathbb{R}$ with

$$\Phi(x) := \int_{-\infty}^{x} \phi(t)\, dt = \frac{1 + \operatorname{erf}\left(\frac{\sqrt{2}x}{2}\right)}{2}. \tag{4}$$

The envelopes of the error function are already available in MC++ as an intrinsic function and consequently the McCormick technique provides envelopes of the CDF (see Fig. 2a in ESI). In contrast, the error function is not available as an intrinsic function in BARON and a closed-form expression does not exist. Thus, a numerical approximation is required for optimization in BARON. Common numerical approximations of the error function are only valid for $x \geq 0$ and use point symmetry of the error function. To overcome this technical difficulty in BARON, a big-M formulation with additional binary and continuous variables is a possible workaround. However, this workaround leads to potentially weaker relaxations (see Sect. 2 in the ESI).

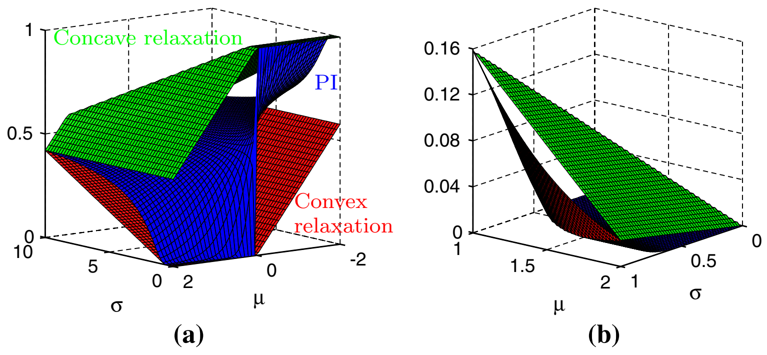


Fig. 2 Graph of the probability of improvement acquisition function (PI) as in Eq. (5) for $f_{\min} = 0$ along with the developed convex and concave relaxations. **a** On the interval $[-2, 2] \times [0, 10]$, the relaxations are constructed on the basis of monotonicity properties of PI. **b** On the interval $[1, 2] \times [0, 1]$, the relaxations are constructed on the basis of componentwise convexity properties via the methods of Meyer and Floudas [56] and Najman et al. [61]. Note that the ranges of μ and σ are different in the two subfigures to highlight the individual relaxations that are derived on different intervals. The ranges in **a** are such that they overlap with all four sets I_1-I_4 defined in Sect. A.2, while the ranges in **b** lie within the set I_4

4.4 Lower confidence bound acquisition function

The lower confidence bound (LCB) (upper confidence bound when considering maximization) is an acquisition function with strong theoretical foundation. For instance, a bound on its cumulative regret, i.e., a convergence rate for Bayesian optimization, for relatively mild assumptions on the black-box function is known [75]. It is given by LCB : $\mathbb{R} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ with

$$\text{LCB}(\mu, \sigma) := \mu - \kappa \cdot \sigma$$

with a parameter $\kappa \in \mathbb{R}_{>0}$. LCB has not been popular in engineering applications as it requires an additional tuning parameter κ and leads to heavy exploration when a rigorous value for κ is chosen [75]. Recently, LCB has gained more popularity through the application as a policy in deep reinforcement learning, e.g., by DeepMind [59]. LCB is a linear function and thus McCormick relaxations are exact.

4.5 Probability of improvement acquisition function

Probability of improvement (PI) computes the probability that a prediction at x is below a given target $f_{\min}$, i.e., $\tilde{\text{PI}}(\boldsymbol{x}) = \mathcal{P}\left(f(\boldsymbol{x}) \leq f_{\min}\right)$. When the underlying function is distributed as a GP with mean μ and variance σ, the PI is given by PI : $\mathbb{R} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ with

$$\text{PI}(\mu, \sigma) := \begin{cases} \Phi\left(\frac{f_{\min} - \mu}{\sigma}\right), & \sigma > 0, \\ 0, & \sigma = 0, \ f_{\min} \leq \mu, \\ 1, & \sigma = 0, \ f_{\min} > \mu. \end{cases} \tag{5}$$

The PI acquisition function is neither convex or concave over its entire domain. However, as analyzed in Sect. A.2, there are parts of its domain over which the function is *componentwise* convex or convex with respect to σ or μ. For componentwise convex or concave functions, there exist methods for constructing tight relaxations. [56] introduced a method for constructing concave relaxations for componentwise convex functions (or vice versa). In particular, the concave envelope of a componentwise convex function is polyhedral [78], and thus the method of [56] amounts to finding the correct combinations of corners of the considered interval box to construct the facets of the polyhedral concave envelope. [61], in contrast, introduce a method for constructing convex relaxations of componentwise convex functions that satisfy a certain monotonicity condition on their first order partial derivatives (or, in case of twice continuously differentiable functions, have mixed second-order partial derivatives with constant sign over the box). For functions that are componentwise convex with respect to some and concave with respect to other variables, [61] also show that by taking the secant with respect to the concave (or convex) variables, one can obtain a relaxation that is componentwise convex (or concave) with respect to all variables. Using the aforementioned methods, this function can then be further relaxed to obtain convex (or concave) relaxations.

We use these methods that exploit componentwise convexity along with techniques that exploit monotonicity properties to construct tight relaxations of the PI acquisition function. The procedure for constructing these relaxations is described in detail in "Appendix A.2". Examples for the resulting relaxations on two subsets of the domain of PI are shown in Fig. 2.

4.6 Expected improvement acquisition function

EI is the acquisition function that is most commonly used in Bayesian optimization [40]. It is defined as $\tilde{\mathrm{EI}}(\boldsymbol{x}) = \mathbb{E}\big[\max(f_{\min} - f(\boldsymbol{x}), 0)\big]$. When the underlying function is distributed as a GP, $\mathrm{EI} : \mathbb{R} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ is given by

$$\mathrm{EI}(\mu, \sigma) := \begin{cases} (f_{\min} - \mu) \cdot \Phi\left(\frac{f_{\min} - \mu}{\sigma}\right) + \sigma \cdot \phi\left(\frac{f_{\min} - \mu}{\sigma}\right), & \sigma > 0 \\ f_{\min} - \mu, & \sigma = 0, \quad \mu < f_{\min} \\ 0 & \sigma = 0, \quad \mu \geq f_{\min} \end{cases} \tag{6}$$

As noted by Jones et al. [40], EI is componentwise monotonic and thus, exact interval bounds can easily be derived. In Sect. A.3, we show that EI is convex and we provide its envelopes. As EI is not available as an intrinsic function in BARON, an algebraic reformulation is necessary that uses Eq. (6) where Φ is substituted from Eq. (4) with Eq. (1) in ESI and ϕ from Eq. (3). In addition, some workaround would be necessary for $\sigma = 0$ (e.g., additional binary variable and big-M formulation).

5 Implementation

The described methods are implemented in our open-source solver MAiNGO [7] and the MeLOn toolbox [71]. The modeling interfaces of MAiNGO (currently either text-based input or a C++ API) allow a convenient implementation of RS models without having to eliminate variables symbolically. Instead, the sequential evaluation of model equations can be expressed as in procedural programming paradigms.

MAiNGO implements a spatial B&B algorithm enhanced with some features for range reduction [32,49,67] and a multi-start heuristic. A directed acyclic graph representation of the model is constructed using the MC++ library [16] and evaluated in different arithmetics: We use automatic differentiation via FADBAD++ [3] to obtain first and second derivatives of functions and provide them to the desired local solver. Currently, MAiNGO supports local solvers found in the NLopt package [39], IPOPT and Knitro. These local solvers can be used for pre-processing and solving the upper bounding problems. In the presented manuscript, we use SLSQP [45] in the pre-processing and in the upper bounding. During pre-processing, a simple multistart heuristic initializes the first local search at the center point of the variable ranges. Subsequent local searches are initialized randomly within the variable ranges.

MAiNGO constructs (multivariate) McCormick relaxations of factorable functions [53,81]. The convex and concave relaxations together with their subgradients [58] are constructed through the MC++ library [16]. The necessary interval extensions are provided through FILIB++ [35]. MAiNGO currently supports CPLEX [38] and CLP [19] as linear programming solvers for lower bounding. In this work, the convex relaxations of the objective and the constraints are linearized at the center point of each node. Subsequently, CPLEX [38] solves the resulting linear problems for lower bounding. MAiNGO can also be run in parallel on multiple cores through MPI. For a fair comparison, we run all optimizations on a single core in this work.

The GP models, acquisition functions, and training scripts are available open-source within the MeLOn toolbox [71] and the relaxations of the corresponding functions are available through the MC++ library used by MAiNGO.

In order to install MAiNGO, please visit our public git repository at https://git.rwth-aachen.de/avt.svt/public/maingo. Our machine learning toolbox MeLOn comes as a submodule of MAiNGO and will be installed with MAiNGO. Currently, MAiNGO can be run using our C++ interface or using our own modeling language called ALE [26]. At the time of writing this manuscript, the authors develop a new Python interface for MAiNGO which will be available soon via pypi.

6 Numerical results

We now investigate the numerical performance of the proposed method on one core of an Intel Xeon CPU with 2.60 GHz, 128 GB RAM and Windows Server 2016 operating system. We present three case studies. We use MAiNGO version v0.2.1 and BARON v19.12.7 through GAMS v30.2.0 to solve different optimization problems with GPs embedded on a single core. Note that we use CPLEX as a lower bounding solver in both BARON and MAiNGO. In MAiNGO we use SLSQP [45] in the pre-processing and in the upper bounding. By default, BARON automatically selects NLP solvers and may switch between different NLP solvers.

First, we illustrate the scaling of the method w.r.t. the number of training data points on a representative test function. Herein, the estimate of the GP is optimized. Second, we consider a chemical engineering case study with a chance constraint, which utilizes the variance prediction of a GP. Third, we optimize an acquisition function that is commonly used in Bayesian optimization on a chemical engineering dataset.

6.1 Illustrative example and scaling of the algorithm

In the first illustrative example, the peaks function is learned by GPs. Then, the GP predictions are optimized on $\tilde{X} = \{x_1, x_2 \in \mathbb{R}\colon -3 \leq x_1, x_2 \leq 3\}$. The peaks function is given by $f_{peaks} : \mathbb{R}^2 \rightarrow \mathbb{R}$ with

$$\begin{aligned} &f_{\text{peaks}}(x_1, x_2) \\ &:= 3\,(1 - x_1)^2 \cdot e^{-x_1^2 - \,(x_2+1)^2} - 10 \cdot \left(\frac{x_1}{5} - x_1^3 - x_2^5\right) \cdot e^{-x_1^2 - x_2^2} - \frac{e^{-(x_1+1)^2 - x_2^2}}{3} \end{aligned}$$

The two-dimensional function has multiple suboptimal local optima and one unique global minimizer at $\boldsymbol{x}^* \approx [0.228, -1.626]^T$ with $f_{\text{peaks}}(\boldsymbol{x}^*) \approx -6.551$.

We generate various training data on $\tilde{X}$ using a Latin hypercube sampling of sizes 10, 20, 30,…, 500. Then, we train GPs with $k_{\nu=1/2}(d)$, $k_{\nu=3/2}(d)$, $k_{\nu=5/2}(d)$, and $k_{SE}(d)$ covariance functions on the data. The parameters of the trained GPs are saved in individual JSON files. After training, the JSON files are read by the solver and the predictions of the GPs are minimized using the RS and FS formulation to locate an approximation of the minimum of f_{peaks}. We run optimizations in MAiNGO once using the developed envelopes and once using standard McCormick relaxations. Due to long CPU times, we run optimizations for the FS formulations only for up to 250 data points in MAiNGO. The whole data generation, training, and optimization procedure are repeated 50 times for each data set. Thus, we train a total of 10, 000 GPs and run 90, 000 optimization problems in MAiNGO. We also solve the FS and RS formulation in BARON by automatically parsing the problem from our C++ implementation to GAMS. This is particularly important in the RS as equations with several thousand characters are generated. We solve the RS problem for up to 360 and the FS for up to 210 data points in BARON due to the high computational effort. The optimality tolerances are set to $\epsilon_{\text{abs. tol.}} = 10^{-3}$ and $\epsilon_{\text{rel. tol.}} = 10^{-3}$ and the maximum CPU time is set to 1, 000 CPU seconds. The reported CPU times do not include any compilation time in MAiNGO and BARON. Note that the MAiNGO code is just compiled once for each problem class because the individual GPs are parameterized by JSON files. Thus, no repeated compilation is necessary. The feasibility tolerances are set to 10^{-6}. The analysis in this section is based on results for the $k_{\nu=5/2}$ covariance function. The detailed results for the other covariance functions show qualitatively similar results (c.f. ESI Sect. 4). Also, the results in this section are based on the median computational times of the 50 repetitions because the variations are comparably small. Boxplots that illustrate the variance are provided in Sect. 4 of the ESI.

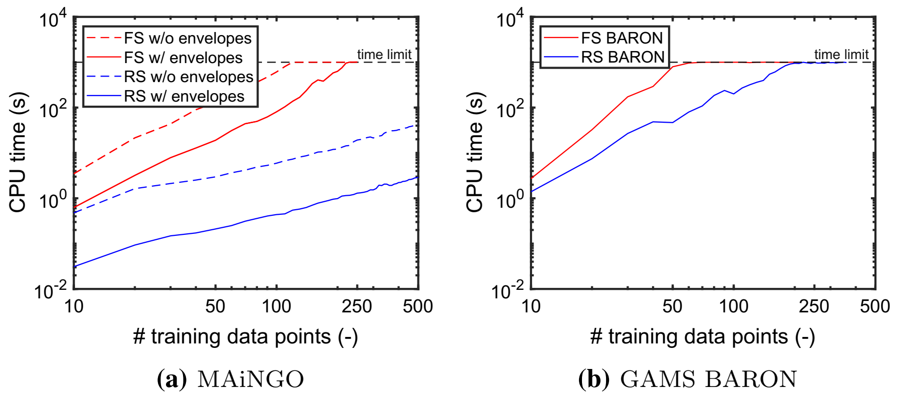


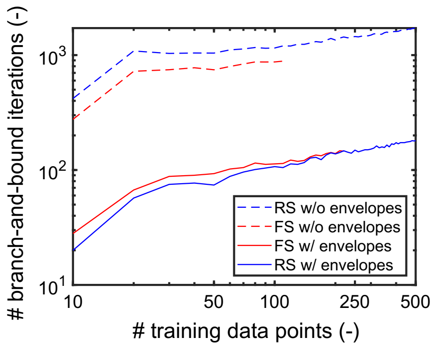
Fig. 3 Comparison of the total CPU time for optimization, i.e., the sum of preprocessing time and B&B time, of GPs with $k_{\nu=5/2}$ covariance function. The plots show the median of 50 repetitions of data generation, GP training, and optimization. Note that #points are incremented in steps of 10 and the lines are interpolations between them

In the FS, this problem has $D + 2 \cdot N + 2$ equality constraints and $2 \cdot D + 2 \cdot N + 2$ optimization variables while the RS has D optimization variables and no equality constraints. Note that for practical applications the number of training data points is usually much larger than the dimension of the inputs, i.e., $N \gg D$. The full problem formulation is also provided in ESI Sect. 3.

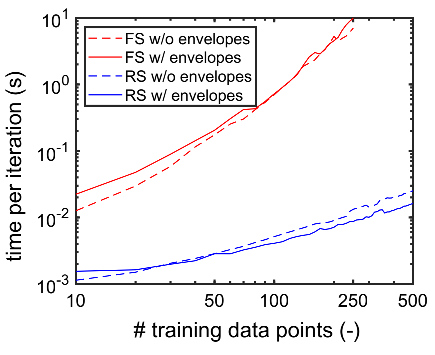
Figure 3 shows a comparison of the CPU time for optimization of GPs. For the solver MAiNGO, Fig. 3a shows that RS formulation outperforms the FS formulation by more than one order of magnitude and shows a more favorable scaling with the number of training data points. For example, the speedup increases to a factor of 778 for 250 data points. Notably, the achieved speedup increases drastically with the number of training data points (c.f. ESI Sect. 4). This is mainly due to the fact that the CPU times for the FS formulations scale approximately cubically with the data points ($\text{CPU}_{FS\ w/\ env}(N) = 1.053 \cdot 10^{-4} N^{2.958}$ sec with $R^2 = 0.993$) while the ones for the RS scale almost linearly ($\text{CPU}_{RS\ w/\ env}(N) = 0.0022 \cdot N^{1.156}$ sec with $R^2 = 0.995$).

In general, the number of optimization variables can lead to an exponential growth of the worst-case B&B iterations and thus runtime. In this particular case, the number of B&B iterations is very similar for the FS and RS formulation (see Fig. 4a). Instead, for the present problems the number of B&B iterations is more influenced by the use of tight relaxations. Figure 4b shows that the CPU time per iteration increases drastically with problem size in the FS while it increases only moderately in the RS. This indicates that the solution time of the lower bounding, upper bounding, and bound tightening subproblems scales favorably in the RS and that this is the main reason for speedup of the RS formulation in MAiNGO. This is probably due to the smaller subproblem sizes when using McCormick relaxations in the RS formulation (c.f. discussion in Sect. 1).

The use of envelopes of covariance functions also improves computational performance (see Fig. 3a). However, this effect is approximately constant over the problem size (c.f. Fig. 3 in ESI Sect. sec:globalspsgpspsoptimizationspsRelaxationsspsofsps RelevantspsFunctionsspsforspsGPsspsandspsBayesianspsOptimization). In other



(a) B&B iterations in MAiNGO



(b) CPU time per B&B iteration in MAiNGO

Fig. 4 Comparison of number of B&B iterations of optimization problems with GPs embedded with $k_{\nu=5/2}$ covariance function. The plots show the median of 50 repetitions of data generation, GP training, and optimization. Note that #points are incremented in steps of 10 and the lines are interpolations between them

words, the CPU time shows a similar trend for the cases with and without envelopes in Fig. 3a. In the RS, the CPU time with envelopes takes on average 7.1% of the CPU time without envelopes ($\approx$ 14 times less). In the FS, the impact of the envelopes is less pronounced, i.e., the CPU time w/ envelopes is on average 15.1% of the CPU time w/o envelopes ($\approx$ 6.6 times less). Figure 4a shows that the envelopes considerably reduce the number of necessary B&B iterations. However, the relaxations do not show a significant influence on the CPU time per iteration (see Fig. 4a).

The results of this numerical example show clearly that the development of tight relaxations is more important for the RS formulation than for the FS. As shown in Sect. 3.4.2 of [4], this effect can be explained by the fact that in RS, it is more likely to have reoccurring nonlinear factors which can cause the McCormick relaxations to become weaker (c.f. also the relationship to the AVM in this case explored in [81]). However, in this study, the improvement in relaxations is outweighed by the increase of CPU time per iteration when additional variables are introduced in the FS.

The RS formulation also performs favorably compared to the FS formulation in the solver BARON (see Fig. 3). However, the differences between the CPU times are less pronounced. In contrast to MAiNGO, the number of B&B iterations in the FS and RS drastically increase with increasing number of training data points when using BARON (c.f. Fig. 4 in ESI). Also, the time per B&B iteration is similar between RS and FS. This is probably due to the AVM method for the construction of relaxations. The AVM method introduces auxiliary variables for some factorable terms. Thus, the size of the subproblems in BARON increases with the number of training data points regardless of which of the two formulations is used.

The results of the optimizations also provide information about the ability of GP surrogate models to approximate a function for optimization. The results show that the solution point of the GP optimization problem approximately converges to the optimum of the learned peaks function for all covariance functions. However, it is clear that some covariance functions lead to more accurate solution for the same number of training data points in this particular case. In the ESI, we provide figures that show the solution point and objective function value over the number of data points for this problem (Figs. 8–11). Interestingly, the objective function value is overestimated considerably for all problems.

6.2 Chance-constrained programming

Probabilistic constraints are relevant in engineering and science [18] and GPs have been used in the previous literature to formulate chance constraints, e.g., in model predictive control [11] or production planning [87].

As a second case study, we consider the N-benzylation reaction of α methylbenzylamine with benzylbromide to form desired secondary (2$\,^\circ$) amine and undesired tertiary (3$\,^\circ$) amine. We utilize an experimental data set consisting of 78 data points from a robotic chemical reaction platform [69]. We aim to maximize the expected space-time yield of 2$\,^\circ$ amine (2$\,^\circ$-STY) and ensure that the probability of a product quality constraint satisfaction is above 95%. The 2$\,^\circ$-STY and yield of 3$\,^\circ$ amine impurity (3$\,^\circ$-Y) are modeled by individual GPs. Thus, we solve optimization problems with two GPs embedded. The chance-constrained optimization problem is formulated as follows

$$\begin{aligned} \min_{\boldsymbol{x}\in E} \quad & -\mathbb{E}\big[f_{\text{STY}}(\boldsymbol{x})\big] \\ \text{s.t.} \quad & \mathcal{P}\left(f_{\text{impurity}}(\boldsymbol{x}) \le c\right) \ge 95\% \end{aligned}$$

Here, the objective is to minimize the negative of the expected STY. This corresponds to minimizing the negative prediction of the GP, i.e., $-m_{\mathcal{D},2\,^\circ-\text{STY}}$. The chance constraint ensures that the impurity is below a parameter c with a probability of 95%. This corresponds to the constraint $m_{\mathcal{D},3^\circ-Y} + 1.96 \cdot \sqrt{k}_{\mathcal{D},3^\circ-Y} \le c$ with $c = 5$.

The optimization is conducted with respect to four optimization variables: (1) the primary (1°) amine flow rate of the feed varying between 0.2 and 0.4 mL min^{-1}, (2) the ratio between benzyl bromide and 1° amine varying between 1.0 and 5.0, (3) the ratio between solvent and 1° amine varying between 0.5 and 1.0, and (4) the temperature varying between 110 and 150 °C.

As this problem is highly multimodal and difficult to solve, we increase the number of local searches in pre-processing in MAiNGO to 500 and increase the maximum CPU time to 24 hours. The computational performance of the different methods is given in Table 1. The results show that none of the considered methods converged to the desired tolerance within the time limit. The RS formulation in MAiNGO that uses the proposed envelopes outperforms the other formulations and BARON solver as it yields the smallest optimality gap. Note that the considered SLSQP solver does not find any valid solution point in the FS in MAiNGO while feasible points are found in the RS. This demonstrates that the RS formulation can also be advantageous for local solvers. Note that when using IPOPT [84] with 500 multistart points in the FS formulation in MAiNGO, it identifies a local optimum with $f^* = -226.5$ in the pre-

Table 1 Numerical results of the N-benzylation reaction optimization with chance constraint (Sect. 6.2)

	Solver	CPU [s]	Iter.	UB	LB	Abs. gap
(FS)	MAiNGO w/ Env.	86,400	26,761	N/A	−3792	N/A
	MAiNGO w/o Env.	86,400	20,795	N/A	−1704.1	N/A
	BARON	86,400	219,239	−226.5	−53,775.0	53,548.5
(RS)	MAiNGO w/ Env.	86,400	$1.6 \cdot 10^6$	−226.5	−244.8	18.3
	MAiNGO w/o Env.	86,400	$1.1 \cdot 10^6$	−226.5	−573.2	346.7
	BARON	86,400	7927	−226.5	−56,394.0	56,167.5

The FS formulation has 330 optimization variables, 326 equality constraints, and 1 inequality constraint. The RS formulation has 4 optimization variables, 0 equality constraints, and 1 inequality constraint. The CPU time limit is 86,400 s

processing. In the ESI, we provide a brief comparison of a few pre-processing settings for this case study.

The best solution of the optimization problem that we found is $x_1 = 0.40\ \text{min}^{-1}$, $x_2 = 1.0$, $x_3 = 0.5$, and $x_4 = 123.5\,°\text{C}$. At the optimal point, the predicted 2 °-STY is 226.5 kg m^{-3} h^-1 with a variance of 17.1 while the predicted amine impurity is 4.2 % with a variance of 0.17. The result shows that the probability constraint ensures a safety margin between the predicted impurity and $c = 5$. Note that the chance constraint is active at the optimal solution point.

6.3 Bayesian optimization

In the third case study, we consider the synthesis of layer-by-layer membranes. Membrane development is a prerequisite for sustainable supply of safe drinking water. However, synthesis of membranes is often based on try-and-error leading to extensive experimental efforts, i.e., building and measuring a membrane in the development phase usually takes several weeks per synthesis protocol. In this case study, we plan to improve the retention of Na_2SO_4 salt of a recently developed layer-by-layer nanofiltration membrane system. The optimization variables are the sodium chloride concentration in the polyelectrolyte solution $c_{NaCl} \in [0, 0.5]$ gL^-1, the deposited polyelectrolyte mass $m_{PE} \in [0, 5]$ gm^{-2}, and the number of layers $N_{layer} \in \{1, 2, 3, ..., 10\}$. The detailed description of the setup is given in the literature [55,65]. Overall, we utilize 63 existing data points from previous literature [55]. We identify a promising synthesis protocol based on the EI acquisition function by solving:

$$\min_{\boldsymbol{x} \in E} \quad -\text{EI}\big(m_{\mathcal{D}}(\boldsymbol{x}), k_{\mathcal{D}}(\boldsymbol{x})\big)$$

with $\boldsymbol{x} = [c_{NaCl}, m_{PE}, N_{layer}]^T$. Thus, this numerical example corresponds to one step of a Bayesian optimization setup for this experiment. Global optimization of the acquisition function is particularly relevant due to inherent multimodality of the

Table 2 Numerical results of the membrane synthesis optimization (Sect. 6.3)

	Solver	CPU [s]	Iter.	UB	LB	Abs. gap
(FS)	MAiNGO w/ Env.	3802	25,995	−2.025	−2.027	0.002
(RS)	MAiNGO w/ Env.	405	14,331	−2.025	−2.027	0.002

The FS formulation has 136 optimization variables and 133 equality constraints. The RS formulation has 3 optimization variables and no equality constraints

acquisition functions [44] and high cost of experiments. Note that the experimental validation of this data point is not within the scope of this work.

The computational performance of the proposed method is summarized in Table 2. Using the solver MAiNGO, the RS formulation converges approximately 9 times faster to the desired tolerance compared to the FS formulation. Herein, we use the derived tailored relaxations of the EI acquisition function and envelopes of the covariance functions in both cases. Notably, the FS requires approximately 1.8 times the number of B&B iterations compared to the RS formulation, which is much less than the overall speedup. Thus, the results are in good agreement with the previous examples showing that the reduction of CPU time per iteration in the RS has a major contribution to the overall speedup. For this example, a comparison to BARON is omitted due to necessary workarounds including several integer variables and function approximations for CDF and EI (c.f., Sects. 4.3, 4.6).

The optimal solution point of the optimization problem is $c_{NaCl} = 0.362$ gL$^{-}1$, $m_{PE} = 0$ gm^{-2}, and $N_{layer} = 4$. The expected retention is 85.32 with a standard deviation $\sigma = 14.8$. The expected retention is actually worse than the best retention in the training data of 96.1. However, Bayesian optimization takes also the high variance of the solution into account, i.e., it is also exploring the space. EI identifies an optimal trade-off between exploration and exploitation.

7 Conclusions

We propose a RS formulation for the deterministic global solution of problems with trained GPs embedded. Also, we derive envelopes of common covariance functions or tight relaxations of acquisition functions leading to tight overall problem relaxations.

The computational performance is demonstrated on illustrative and engineering case studies using our open-source global solver MAiNGO. The results show that the number of optimization variables and equality constraints are reduced significantly compared to the FS formulation. In particular, the RS formulation results in smaller subproblems whose size does not scale with the number of training data points when using McCormick relaxations. This leads to tractable solution times and overcomes previous computational limitations. For example, we archive a speedup factor of 778 for a GP trained on 250 data points. The GP training methods and models are provided as an open-source module called "MeLOn—**M**achin**e** **L**earning Models for **O**ptimizatio**n**" toolbox [71].

We thus demonstrate a high potential for future research and industrial applications. For instance, global optimization of the acquisition function can improve the efficiency of Bayesian optimization in various applications. It also allows to easily include integer decisions and nonlinear constraints in Bayesian optimization. Furthermore, the proposed method could be extended to various related concepts such as multi-task GPs [8], deep GPs [21], global model-predictive control with dynamic GPs [13,85], and Thompson sampling [12,17]. Finally, the proposed work demonstrates that the RS formulation may be advantageous for a wide variety of problems that have a similar structure, including various machine-learning models, model ensembles, Monte-Carlo simulation, and two-stage stochastic programming problems.

Acknowledgements We are grateful to Benoît Chachuat for providing MC++. We are grateful to Daniel Menne, Deniz Rall and Matthias Wessling for providing the experimental membrane data for Example 3. Finally, we thank the area editor, the associate editor, the technical editor, and the anonymous reviewers for their valuable comments and suggestions which helped us to improve this manuscript.

Author contributions AMS designed the research concept, ran simulations, and wrote the manuscript. AMS and DB interpreted the results. AMS, DB, JN, and AM edited the manuscript. AMS, XL, DG implemented the GP models and parser. AMS, DB, JN and TK derived the function relaxations. JN and DB implemented the function relaxations. AM is principle investigator and guided the effort.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

A Derivations of convex and concave relaxations

For the sake of simplicity, we use the same symbols in each subsection for the corresponding convex (F^{cv}) and concave (F^{cc}) relaxations. To solve the one-dimensional nonlinear equations that arise multiple times in the following, we use Newton's method with 100 iterations and a tolerance of 10^{-9}. If this is not successful, we run a golden section search as a backup.

A.1 Probability density function of Gaussian distribution

In this subsection, the envelopes of the PDF are derived on a compact interval $D = [x^L, x^U]$. As the probability density function is one-dimensional, McCormick [53] gives a method to construct its envelopes. The PDF is convex on $]-\infty, -1]$ and

$[1, \infty[$ and it is concave on $[-1, 1]$. Its convex envelope, $F^{cv} : \mathbb{R} \to \mathbb{R}$, and concave envelope, $F^{cc} : \mathbb{R} \to \mathbb{R}$, are given by

$$F^{cv}(x) = \begin{cases} \phi(x), & x^U \le -1, \\ F_2^{cv}(x), & x^L \le -1, \quad -1 \le x^U \le 1, \\ \text{sct}(x), & -1 \le x^L, \quad x^U \le 1, \\ F_4^{cv}(x), & -1 \le x^L, \quad x^U \ge 1, \\ F_5^{cv}(x), & x^L \le -1, \quad x^U \ge 1, \\ \phi(x), & x^L \ge 1 \end{cases}$$

$$F^{cc}(x) = \begin{cases} \text{sct}(x), & x^U \le -1, \\ F_2^{cc}(x), & x^L \le -1, \quad -1 \le x^U \le 1, \\ \phi(x), & -1 \le x^L, \quad x^U \le 1, \\ F_4^{cc}(x), & -1 \le x^L \le 1, \quad x^U \ge 1, \\ F_5^{cc}(x), & x^L \le -1, \quad x^U \ge 1, \\ \text{sct}(x), & x^L \ge 1 \end{cases}$$

where $\text{sct}(x) = \frac{\phi(x^U)-\phi(X^L)}{x^U-x^L} \cdot (x - x^L) + \phi(x^L)$. $F_2^{cc} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_2^{cc}(x) = \begin{cases} \frac{\phi(x_{c,2}^U)-\phi(x^L)}{x_{c,2}^U-x^L} \cdot (x - x^L) + \phi(x^L), & x \le x_{c,2}^U, \\ \phi(x), & x > x_{c,2}^U, \end{cases}$$

where $x_{c,2}^U = \min(x_{c,2}^{U*}, x^U)$ and $x_{c,2}^{U*}$ is the solution of $\left.\frac{d\phi}{dx}\right|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}, \quad x \in [-1, 0]$. $F_2^{cv} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_2^{cv}(x) = \begin{cases} \phi(x), & x \le x_{c,2}^L, \\ \frac{\phi(x^U-\phi(x_{c,2}^L))}{x^U-x_{c,2}^L} \cdot (x - x_{c,2}^L) + \phi(x_{c,2}^L), & x > x_{c,2}^L, \end{cases}$$

where $x_{c,2}^L = \max(x_{c,2}^{L*}, x^L)$ and $x_{c,2}^{L*}$ is the solution of $\left.\frac{d\phi}{dx}\right|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}, \quad x \in [x^L, -1]$. $F_4^{cc} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_4^{cc}(x) = \begin{cases} \phi(x), & x < x_{c,4}^L, \\ \frac{\phi(x^U)-\phi(x_{c,4}^L)}{x^U-x_{c,4}^L} \cdot (x - x_{c,4}^L) + \phi(x_{c,4}^L), & x \ge x_{c,4}^L, \end{cases}$$

where $x_{c,4}^L = \max(x_{c,4}^{L*}, x^L)$ and $x_{c,4}^{L*}$ is the solution of $\frac{d\phi}{dx}\Big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}$, $x \in [0, x^U]$. $F_4^{cv} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_4^{cv}(x) = \begin{cases} \frac{\phi(x_{c,4}^U)-\phi(x_L)}{x_{c,4}^U - x_L} \cdot (x - x_L) + \phi(x_L), & x \le x_{c,4}^U, \\ \phi(x), & x > x_{c,4}^U, \end{cases}$$

where $x_{c,4}^U = \min(x_{c,4}^{U*}, x^U)$ and $x_{c,4}^{U*}$ is the solution of $\frac{d\phi}{dx}\Big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}$ on $[x^L, -1]$ if $x^L + x^U < 0$ or $[1, x^U]$ if $x^L + x^U > 0$. The case $x^L + x^U = 0$ is symmetrical and handled separately to avoid numerical issues in Newton. $F_5^{cc} : \mathbb{R} \to \mathbb{R}$ is given by a combination of F_2^{cc} and F_4^{cc}:

$$F_5^{cc}(x) = \begin{cases} \frac{\phi(x_{c,2}^U)-\phi(x^L)}{x_{c,2}^U - x^L} \cdot (x - x^L) + \phi(x^L), & x \le x_{c,2}^U, \\ \phi(x), & x_{c,2}^U < x < x_{c,4}^L, \\ \frac{\phi(x^U)-\phi(x_{c,4}^L)}{x^U - x_{c,4}^L} \cdot (x - x_{c,4}^L) + \phi(x_{c,4}^L), & x \ge x_{c,4}^L, \end{cases}$$

$F_5^{cv} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_5^{cv}(x) = \begin{cases} \frac{\phi(x_{c,5}^U)-\phi(x^L)}{x_{c,5}^U - x^L} \cdot (x - x^L) + \phi(x^L), & x^L + x^U \ge 0, \quad x \le x_{c,5}^U, \\ \phi(x), & x^L + x^U \ge 0, \quad x > x_{c,5}^U, \\ \phi(x), & x^L + x^U < 0, \quad x \le x_{c,5}^L, \\ \frac{\phi(\phi(x^U - x_{c,5}^L))}{x^U - x_{c,5}^L} \cdot (x - x_{c,5}^L) + \phi(x_{c,5}^L), & x^L + x^U < 0, \quad x > x_{c,5}^L, \end{cases}$$

where $x_{c,5}^U = \min(x_{c,5}^{U*}, x^U)$ and $x_{c,5}^{U*}$ is the solution of $\frac{d\phi}{dx}\Big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}$ on $[x^L, 0]$. Further, $x_{c,5}^L = \max(x_{c,5}^{L*}, x^L)$ and $x_{c,5}^{L*}$ is the solution of $\frac{d\phi}{dx}\Big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}$ on $[0, x^U]$.

A.2 Probability of improvement acquisition function

In this section, a tight relaxation of the PI acquisition function is derived. PI is continuous for all $(\mu, \sigma) \in \mathbb{R} \times [0, \infty[\ \setminus \{(0, 0)\}$, since $\lim_{x\to+\infty} \Phi(x) = 1$ and $\lim_{x\to-\infty} \Phi(x) = 0$.

A.2.1 Monotonicity

From the gradient of PI on $\mathbb{R} \times (0, \infty)$,

$$\nabla \mathrm{PI}(\mu, \sigma) = -\frac{1}{\sigma^2 \cdot \sqrt{2\pi}} \cdot \exp\left(-\frac{(f_{\min} - \mu)^2}{2 \cdot \sigma^2}\right) \begin{bmatrix} \sigma \\ (f_{\min} - \mu) \end{bmatrix},$$

Table 3 componentwise convexity properties of PI over subsets of its domain

Subset	componentwise property	
I_1	Concave w.r.t. μ	Convex w.r.t. σ
I_2	Convex w.r.t. μ	Concave w.r.t. σ
I_3	Concave w.r.t. μ	Concave w.r.t. σ
I_4	Convex w.r.t. μ	Convex w.r.t. σ

where $f_{\min}$ is a given target, we identify the following monotonicity properties:

- PI is monotonically decreasing with respect to μ.
- If $\mu < f_{\min}$ then PI is monotonically decreasing with respect to σ.
- If $\mu \geq f_{\min}$ then PI is monotonically increasing with respect to σ (recall that $\mathrm{PI}(f_{\min}, 0) = 0$, and $\mathrm{PI}(f_{\min}, \sigma) = 0.5\ \forall \sigma \in\ ]0, \infty[\ $).

These properties can be used to obtain exact interval bounds on the function values of PI. Furthermore, they can be exploited to construct relaxations as described in Sect. A.2.3.

A.2.2 Componentwise convexity

The Hessian of PI on $\mathbb{R} \times (0, \infty[$ is given by

$$\nabla^2 \mathrm{PI}(\mu, \sigma) = \begin{bmatrix} -\frac{f_{\min}-\mu}{\sigma^3} & -\frac{(f_{\min}-\mu)^2-\sigma^2}{\sigma^4} \\ -\frac{(f_{\min}-\mu)^2-\sigma^2}{\sigma^4} & \frac{(f_{\min}-\mu)\cdot\left(2\sigma^2-(f_{\min}-\mu)^2\right)}{\sigma^5} \end{bmatrix} \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(f_{\min}-\mu)^2}{2\sigma^2}}.$$

The Hessian is indefinite and PI is therefore neither convex nor concave on its whole domain. However, we find *componentwise convexity* properties on certain parts of the domain, i.e., convexity with respect to one variable when the other is fixed. To this end, we divide the domain into the following four sets:

- $I_1 := \{(\mu, \sigma) \mid \mu \leq f_{\min} \wedge \mu - f_{\min} \geq -\sqrt{2}\sigma\}$,
- $I_2 := \{(\mu, \sigma) \mid \mu \geq f_{\min} \wedge \mu - f_{\min} \leq +\sqrt{2}\sigma\}$,
- $I_3 := \{(\mu, \sigma) \mid \mu \leq f_{\min} \wedge \mu - f_{\min} \leq -\sqrt{2}\sigma\}$,
- $I_4 := \{(\mu, \sigma) \mid \mu \geq f_{\min} \wedge \mu - f_{\min} \geq +\sqrt{2}\sigma\}$.

On these sets, PI has the componentwise convexity properties listed in Table 3.

A.2.3 Relaxations

We construct relaxations of PI over a given subset $\mathcal{X} = [\mu^{\mathrm{L}}, \mu^{\mathrm{U}}] \times [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}]$ of its domain depending on which of the four sets I_1-I_4 contains the set $\mathcal{X}$. If $\mathcal{X} \subset I_1 \cup I_2$, we use the McCormick relaxations obtained by applying the multivariate composition theorem [81] to the composition of the rational function $\frac{f_{\min}-\mu}{\sigma}$ with Φ (c.f. Eq. (5)), since these are already very tight. If $\mathcal{X}$ does not fully lie within $I_1 \cup I_2$, the McCormick relaxations get increasingly weaker and we thus resort to other methods as described in the following.

If $\mathcal{X} \subset I_4$, PI is componentwise convex with respect to both variables. Therefore, the concave envelope of PI over $\mathcal{X}$ consists of two planes anchored at the four corner points of $\mathcal{X}$ and can be calculated as described by [56]. A tight convex relaxation can be obtained using the method by [61]. Since the off-diagonal entries of the Hessian have a constant sign over I_4, a sufficient condition for this method is fulfilled (c.f. Corollary 1 in [61]). An example for the resulting relaxation is shown in Fig. 2b. Similarly, if $\mathcal{X} \subset I_3$, PI is componentwise concave and we obtain its convex envelope using the method by [56] and a tight concave relaxation using the method by [61].

If $\mathcal{X} \subset I_2 \cup I_4$, we construct relaxations exploiting the monotonicity properties of PI. Since for all $(\mu, \sigma) \in I_2 \cup I_4$ we have $\mu \geq f_{\min}$, PI is thus monotonically decreasing in μ and increasing in σ over $\mathcal{X}$. Therefore, we can construct a convex relaxation $\mathrm{PI}^{\mathrm{cv}}_{2,4} : \mathcal{X} \to [0, 1]$ as

$$\mathrm{PI}^{\mathrm{cv}}_{2,4}(\mu, \sigma) := \max\left(f^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}(\mu),\, f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}(\sigma)\right), \tag{7}$$

where $f^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}$ and $f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}$ are the convex envelopes of the univariate functions

$$f_{\sigma^{\mathrm{L}}} : [\mu^{\mathrm{L}}, \mu^{\mathrm{U}}] \to [0, 1], \mu \mapsto \mathrm{PI}(\mu, \sigma^{\mathrm{L}})$$

and

$$f_{\mu^{\mathrm{U}}} : [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}] \to [0, 1], \sigma \mapsto \mathrm{PI}(\mu^{\mathrm{U}}, \sigma), \tag{8}$$

respectively, i.e., they correspond to the function PI restricted to one-dimensional facets of $\mathcal{X}$ at σ^{L} and μ^{U}. Both $f^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}(\mu)$ and $f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}(\sigma)$ are valid relaxations of PI because of the monotonicity of PI over $I_2 \cup I_4$. By taking the pointwise maximum in (7), we obtain a tighter relaxation while preserving convexity. To compute $f^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}$ and $f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}$, we can use the method described in Sect. 4 of [53] because they are one-dimensional functions with a known inflection point. To apply this method, we typically need to solve a one-dimensional nonlinear equation, which we do via Newton's method. A concave relaxation can be obtained analogously using concave envelopes of PI over one-dimensional facets of $\mathcal{X}$ at σ^{U} and μ^{L}. If $\mathcal{X} \subset I_1 \cup I_3$, an analogous method can be used since PI is monotonically increasing in both μ and σ.

In the most general case, $\mathcal{X}$ contains parts of all four sets I_1-I_4. In this case, we can still obtain relaxations by exploiting monotonicity properties. In particular, we compute a convex relaxation $\mathrm{PI}^{\mathrm{cv}}_{1-4} : \mathcal{X} \to [0, 1]$ as

$$\mathrm{PI}^{\mathrm{cv}}_{1-4}(\mu, \sigma) := \max\left(\tilde{f}^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}(\mu),\, f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}(\sigma)\right), \tag{9}$$

where $f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}$ is again the convex relaxation of the univariate function $f_{\mu^{\mathrm{U}}}$ as in (8), which is still valid because PI is decreasing with respect to μ on its entire domain. In contrast, the convex relaxation of the univariate function at σ^{L} as in (7) is not valid because PI is not monotonic with respect to σ. Instead, in (9) it is replaced by the convex relaxation $\tilde{f}^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}$ of the univariate function $\tilde{f}_{\sigma^{\mathrm{L}}} : [\mu^{\mathrm{L}}, \mu^{\mathrm{U}}] \to [0, 1]$ with

$$\tilde{f}_{\sigma^{\mathrm{L}}}(\mu) := \begin{cases} \mathrm{PI}(\mu, \sigma^{\mathrm{L}}), & \mu \geq f_{\min}, \\ \mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) + \frac{\mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) - \mathrm{PI}(\mu^{\mathrm{L}}, \sigma^{\mathrm{U}})}{f_{\min} - \mu^{\mathrm{L}}} (\mu - f_{\min}), & \text{otherwise.} \end{cases} \tag{10}$$

To see that $\tilde{f}^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}$ is a valid relaxation of PI, we first note that by definition it is a relaxation of $\tilde{f}_{\sigma^{\mathrm{L}}}$, so it suffices to show that $\tilde{f}_{\sigma^{\mathrm{L}}}$ is in turn a relaxation of PI. The latter is established in the following Lemma.

Lemma 1 *Let* PI *be defined as in* (5) *and* $\tilde{f}_{\sigma^{\mathrm{L}}}$ *as in* (10). *Then* $\tilde{f}_{\sigma^{\mathrm{L}}}(\mu) \leq \mathrm{PI}(\mu, \sigma)\ \forall (\mu, \sigma) \in \mathcal{X} := [\mu^{\mathrm{L}}, \mu^{\mathrm{U}}] \times [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}]$.

Proof Consider first any fixed $\hat{\mu}$ such that $\hat{\mu} \geq f_{\min}$. In this case, we have $\tilde{f}_{\sigma^{\mathrm{L}}}(\hat{\mu}) = \mathrm{PI}(\hat{\mu}, \sigma^{\mathrm{L}}) \leq \mathrm{PI}(\hat{\mu}, \sigma)\ \forall \sigma \in [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}]$ because of the monotonicity w.r.t σ (c.f. Sect. A.2.1). Next, consider any $\tilde{\mu}$ such that $\tilde{\mu} < f_{\min}$. Note that this implies $\mu^{\mathrm{L}} < f_{\min}$. In this case, we have

$$\begin{aligned} \tilde{f}_{\sigma^{\mathrm{L}}}(\tilde{\mu}) &= \mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) + \frac{\mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) - \mathrm{PI}(\mu^{\mathrm{L}}, \sigma^{\mathrm{U}})}{f_{\min} - \mu^{\mathrm{L}}} (\tilde{\mu} - f_{\min}) \\ &= \mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) \frac{\tilde{\mu} - \mu^{\mathrm{L}}}{f_{\min} - \mu^{\mathrm{L}}} + \mathrm{PI}(\mu^{\mathrm{L}}, \sigma^{\mathrm{U}}) \frac{f_{\min} - \tilde{\mu}}{f_{\min} - \mu^{\mathrm{L}}} \\ &\leq \mathrm{PI}(f_{\min}, \sigma^{\mathrm{U}}) \frac{\tilde{\mu} - \mu^{\mathrm{L}}}{f_{\min} - \mu^{\mathrm{L}}} + \mathrm{PI}(\mu^{\mathrm{L}}, \sigma^{\mathrm{U}}) \frac{f_{\min} - \tilde{\mu}}{f_{\min} - \mu^{\mathrm{L}}} \\ &\leq \mathrm{PI}(\tilde{\mu}, \sigma^{\mathrm{U}}) \\ &\leq \mathrm{PI}(\tilde{\mu}, \sigma)\ \forall \sigma \in [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}], \end{aligned}$$

where the inequalities follow, in this order, from the monotonicity of PI with respect to σ for $\mu \geq f_{\min}$, its componentwise concavity with respect to μ for $\mu < f_{\min}$, and its monotonicity with respect to σ for $\mu < f_{\min}$. □

A.3 Expected improvement acquisition function

We now show that the EI acquisition function is convex. From the Hessian matrix of EI on $\mathbb{R} \times (0, \infty)$

$$\nabla^2 \mathrm{EI}(\mu, \sigma) = \begin{bmatrix} \frac{1}{\sigma} & -\frac{\mu - f_{\min}}{\sigma^2} \\ -\frac{\mu - f_{\min}}{\sigma^2} & \frac{(\mu - f_{\min})^2}{\sigma^3} \end{bmatrix} \cdot \phi\left(-\frac{\mu - f_{\min}}{\sigma}\right),$$

we find the eigenvalues 0 and $\frac{(\mu - f_{\min})^2 + \sigma^2}{\sigma^3} \cdot \phi\left(-\frac{\mu - f_{\min}}{\sigma}\right)$. As $\sigma \geq 0$, $\mathrm{EI}(\cdot, \cdot)$ is convex and the envelopes can be constructed directly.

References

1. Amar, Y., Schweidtmann, A.M., Deutsch, P., Cao, L., Lapkin, A.: Machine learning and molecular descriptors enable rational solvent selection in asymmetric catalysis. Chem. Sci. **10**(27), 6697–6706 (2019). https://doi.org/10.1039/C9SC01844A
2. Androulakis, I.P., Maranas, C.D., Floudas, C.A.: αBB: a global optimization method for general constrained nonconvex problems. J. Glob. Optim. **7**(4), 337–363 (1995). https://doi.org/10.1007/BF01099647
3. Bendtsen, C., Stauning, O.: Fadbad++ (version 2.1): a flexible C++ package for automatic differentiation (2012)
4. Bongartz, D.: Deterministic global flowsheet optimization for the design of energy conversion processes. Ph.D. Thesis, RWTH Aachen University (2020). https://doi.org/10.18154/RWTH-2020-06052
5. Bongartz, D., Mitsos, A.: Deterministic global optimization of process flowsheets in a reduced space using McCormick relaxations. J. Glob. Optim. **20**(9), 419 (2017). https://doi.org/10.1007/s10898-017-0547-4
6. Bongartz, D., Mitsos, A.: Deterministic global flowsheet optimization: between equation-oriented and sequential-modular methods. AIChE J. **65**(3), 1022–1034 (2019). https://doi.org/10.1002/aic.16507
7. Bongartz, D., Najman, J., Sass, S., Mitsos, A.: MAiNGO: McCormick-based Algorithm for mixed integer Nonlinear Global Optimization. Technical report, Process Systems Engineering (AVT.SVT), RWTH Aachen University (2018). http://permalink.avt.rwth-aachen.de/?id=729717
8. Bonilla, E.V., Chai, K.M., Williams, C.: Multi-task Gaussian process prediction. In: Advances in neural information processing systems, pp. 153–160 (2008)
9. Boukouvala, F., Floudas, C.A.: Argonaut: algorithms for global optimization of constrained grey-box computational problems. Optim. Lett. **11**(5), 895–913 (2017). https://doi.org/10.1007/s11590-016-1028-2
10. Boukouvala, F., Misener, R., Floudas, C.A.: Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. Eur. J. Oper. Res. **252**(3), 701–727 (2016). https://doi.org/10.1016/j.ejor.2015.12.018
11. Bradford, E., Imsland, L., Zhang, D., Chanona, E.A.d.R.: Stochastic data-driven model predictive control using Gaussian processes. arXiv:1908.01786 (2019)
12. Bradford, E., Schweidtmann, A.M., Lapkin, A.: Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm. J. Glob. Optim. **71**(2), 407–438 (2018). https://doi.org/10.1007/s10898-018-0609-2
13. Bradford, E., Schweidtmann, A.M., Zhang, D., Jing, K., del Rio-Chanona, E.A.: Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate Gaussian processes. Comput. Chem. Eng. **118**, 143–158 (2018). https://doi.org/10.1016/j.compchemeng.2018.07.015
14. Caballero, J.A., Grossmann, I.E.: An algorithm for the use of surrogate models in modular flowsheet optimization. AIChE J. **54**(10), 2633–2650 (2008). https://doi.org/10.1002/aic.11579
15. Caballero, J.A., Grossmann, I.E.: Rigorous flowsheet optimization using process simulators and surrogate models. In: Computer Aided Chemical Engineering, vol. 25, pp. 551–556. Elsevier (2008)
16. Chachuat, B., Houska, B., Paulen, R., Peric, N., Rajyaguru, J., Villanueva, M.E.: Set-theoretic approaches in analysis, estimation and control of nonlinear systems. IFAC-PapersOnLine **48**(8), 981–995 (2015). https://doi.org/10.1016/j.ifacol.2015.09.097
17. Chapelle, O., Li, L.: An empirical evaluation of Thompson sampling. In: J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 24, pp. 2249–2257. Curran Associates, Inc. (2011). http://papers.nips.cc/paper/4321-an-empirical-evaluation-of-thompson-sampling.pdf
18. Charnes, A., Cooper, W.W.: Chance-constrained programming. Manag. Sci. **6**(1), 73–79 (1959). https://doi.org/10.1287/mnsc.6.1.73
19. CLP, C.O.: Linear programming solver: an open source code for solving linear programming problems (2011). https://doi.org/10.5281/zenodo.3748677
20. Cozad, A., Sahinidis, N.V., Miller, D.C.: Learning surrogate models for simulation-based optimization. AIChE J. **60**(6), 2211–2227 (2014). https://doi.org/10.1002/aic.14418
21. Damianou, A., Lawrence, N.: Deep Gaussian processes. In: Artificial Intelligence and Statistics, pp. 207–215 (2013)
22. Davis, E., Ierapetritou, M.: A Kriging method for the solution of nonlinear programs with black-box functions. AIChE J. **53**(8), 2001–2012 (2007). https://doi.org/10.1002/aic.11228

23. Davis, E., Ierapetritou, M.: A kriging-based approach to MINLP containing black-box models and noise. Ind. Eng. Chem. Res. **47**(16), 6101–6125 (2008). https://doi.org/10.1021/ie800028a
24. Davis, E., Ierapetritou, M.: A centroid-based sampling strategy for Kriging global modeling and optimization. AIChE J. **56**(1), 220–240 (2010). https://doi.org/10.1002/aic.11881
25. Del Rio-Chanona, E.A., Cong, X., Bradford, E., Zhang, D., Jing, K.: Review of advanced physical and data-driven models for dynamic bioprocess simulation: case study of algae-bacteria consortium wastewater treatment. Biotechnol. Bioeng. **116**(2), 342–353 (2019). https://doi.org/10.1002/bit.26881
26. Djelassi, H., Mitsos, A.: libALE—a library for algebraic logical expression trees (2019). https://git.rwth-aachen.de/avt.svt/public/libale. Accessed 8 Nov 2019
27. Eason, J.P., Biegler, L.T.: A trust region filter method for glass box/black box optimization. AIChE J. **62**(9), 3124–3136 (2016). https://doi.org/10.1002/aic.15325
28. Epperly, T.G.W., Pistikopoulos, E.N.: A reduced space branch and bound algorithm for global optimization. J. Glob. Optim. **11**(3), 287–311 (1997). https://doi.org/10.1023/A:1008212418949
29. Freier, L., Hemmerich, J., Schöler, K., Wiechert, W., Oldiges, M., von Lieres, E.: Framework for Kriging-based iterative experimental analysis and design: optimization of secretory protein production in corynebacterium glutamicum. Eng. Life Sci. **16**(6), 538–549 (2016). https://doi.org/10.1002/elsc.201500171
30. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005). https://doi.org/10.1137/S0036144504446096
31. Glassey, J., Von Stosch, M.: Hybrid Modeling in Process Industries. CRC Press (2018)
32. Gleixner, A.M., Berthold, T., Müller, B., Weltge, S.: Three enhancements for optimization-based bound tightening. J. Glob. Optim. **67**(4), 731–757 (2017). https://doi.org/10.1007/s10898-016-0450-4
33. Hasan, M.F., Baliban, R.C., Elia, J.A., Floudas, C.A.: Modeling, simulation, and optimization of postcombustion CO_2 capture for variable feed concentration and flow rate. 2. pressure swing adsorption and vacuum swing adsorption processes. Ind. Eng. Chem. Res. **51**(48), 15665–15682 (2012). https://doi.org/10.1021/ie301572n
34. Helmdach, D., Yaseneva, P., Heer, P.K., Schweidtmann, A.M., Lapkin, A.A.: A multiobjective optimization including results of life cycle assessment in developing biorenewables-based processes. ChemSusChem **10**(18), 3632–3643 (2017). https://doi.org/10.1002/cssc.201700927
35. Hofschuster, W., Krämer, W.: FILIB++ interval library (version 3.0.2) (1998)
36. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 3 edn. Springer, Berlin (1996). https://doi.org/10.1007/978-3-662-03199-5
37. Hüllen, G., Zhai, J., Kim, S.H., Sinha, A., Realff, M.J., Boukouvala, F.: managing uncertainty in data-driven simulation-based optimization. Comput. Chem. Eng. (2019). https://doi.org/10.1016/j.compchemeng.2019.106519
38. International Business Machies: IBM ilog CPLEX (version 12.1) (2009)
39. Johnson, S.G.: The NLopt nonlinear-optimization package (version 2.4.2) (2016)
40. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998). https://doi.org/10.1023/A:1008306431147
41. Kahrs, O., Marquardt, W.: The validity domain of hybrid models and its application in process optimization. Chem. Eng. Process. Process Intensif. **46**(11), 1054–1066 (2007). https://doi.org/10.1016/j.cep.2007.02.031
42. Keßler, T., Kunde, C., McBride, K., Mertens, N., Michaels, D., Sundmacher, K., Kienle, A.: Global optimization of distillation columns using explicit and implicit surrogate models. Chem. Eng. Sci. **197**, 235–245 (2019). https://doi.org/10.1016/j.ces.2018.12.002
43. Keßler, T., Kunde, C., Mertens, N., Michaels, D., Kienle, A.: Global optimization of distillation columns using surrogate models. SN Appl. Sci. **1**(1), 11 (2019). https://doi.org/10.1007/s42452-018-0008-9
44. Kim, J., Choi, S.: On local optimizers of acquisition functions in Bayesian optimization. arXiv:1901.08350 (2019)
45. Kraft, D.: Algorithm 733: TOMP-fortran modules for optimal control calculations. ACM Trans. Math. Softw. (TOMS) **20**(3), 262–281 (1994). https://doi.org/10.1145/192115.192124
46. Krige, D.G.: A statistical approach to some basic mine valuation problems on the witwatersrand. J. South. Afr. Inst. Min. Metall. **52**(6), 119–139 (1951)
47. Liberti, L., Cafieri, S., Tarissan, F.: Reformulations in mathematical programming: a computational approach. In: A. Abraham, A. Hassanien, P. Siarry, A. Engelbrecht (eds.) Foundations of Computational Intelligence Volume 3. Studies in Computational Intelligence, vol. 203, pp. 153–234. Springer, Berlin (2009)

48. Lin, Z., Wang, J., Nikolakis, V., Ierapetritou, M.: Process flowsheet optimization of chemicals production from biomass derived glucose solutions. Comput. Chem. Eng. **102**, 258–267 (2017). https://doi.org/10.1016/j.compchemeng.2016.09.012
49. Locatelli, M., Schoen, F. (eds.): Global Optimization: Theory, Algorithms, and Applications. MOS-SIAM Series on Optimization. Mathematical Programming Society, Philadelphia (2013)
50. Maher, S.J., Fischer, T., Gally, T., Gamrath, G., Gleixner, A., Gottwald, R.L., Hendel, G., Koch, T., Lübbecke, M.E., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Weninger, D., Witt, J.T., Witzig, J.: The SCIP optimization suite (version 4.0)
51. McBride, K., Kaiser, N.M., Sundmacher, K.: Integrated reaction-extraction process for the hydroformylation of long-chain alkenes with a homogeneous catalyst. Comput. Chem. Eng. **105**, 212–223 (2017). https://doi.org/10.1016/j.compchemeng.2016.11.019
52. McBride, K., Sundmacher, K.: Overview of surrogate modeling in chemical process engineering. Chem. Ingenieur Tech. **91**(3), 228–239 (2019). https://doi.org/10.1002/cite.201800091
53. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: part I—convex underestimating problems. Math. Program. **10**(1), 147–175 (1976). https://doi.org/10.1007/BF01580665
54. Mehrian, M., Guyot, Y., Papantoniou, I., Olofsson, S., Sonnaert, M., Misener, R., Geris, L.: Maximizing neotissue growth kinetics in a perfusion bioreactor: an in silico strategy using model reduction and Bayesian optimization. Biotechnol. Bioeng. **115**(3), 617–629 (2018). https://doi.org/10.1002/bit.26500
55. Menne, D., Kamp, J., Wong, J.E., Wessling, M.: Precise tuning of salt retention of backwashable polyelectrolyte multilayer hollow fiber nanofiltration membranes. J. Membr. Sci. **499**, 396–405 (2016). https://doi.org/10.1016/j.memsci.2015.10.058
56. Meyer, C.A., Floudas, C.A.: Convex envelopes for edge-concave functions. Math. Program. **103**(2), 207–224 (2005). https://doi.org/10.1007/s10107-005-0580-9
57. Misener, R., Floudas, C.A.: ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. J. Glob. Optim. **59**(2), 503–526 (2014). https://doi.org/10.1007/s10898-014-0166-2
58. Mitsos, A., Chachuat, B., Barton, P.I.: McCormick-based relaxations of algorithms. SIAM J. Optim. **20**(2), 573–601 (2009). https://doi.org/10.1137/080717341
59. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv:1312.5602 (2013)
60. Mogk, G., Mrziglod, T., Schuppert, A.: Application of hybrid models in chemical industry. In: Computer Aided Chemical Engineering, vol. 10, pp. 931–936. Elsevier (2002). https://doi.org/10.1016/S1570-7946(02)80183-3
61. Najman, J., Bongartz, D., Mitsos, A.: Convex relaxations of componentwise convex functions. Comput. Chem. Eng. **130**, 106527 (2019). https://doi.org/10.1016/j.compchemeng.2019.106527
62. Najman, J., Mitsos, A.: On tightness and anchoring of McCormick and other relaxations. J. Glob. Optim. (2017). https://doi.org/10.1007/s10898-017-0598-6
63. Quirante, N., Javaloyes, J., Caballero, J.A.: Rigorous design of distillation columns using surrogate models based on Kriging interpolation. AIChE J. **61**(7), 2169–2187 (2015). https://doi.org/10.1002/aic.14798
64. Quirante, N., Javaloyes, J., Ruiz-Femenia, R., Caballero, J.A.: Optimization of chemical processes using surrogate models based on a Kriging interpolation. In: Computer Aided Chemical Engineering, vol. 37, pp. 179–184. Elsevier (2015). https://doi.org/10.1016/B978-0-444-63578-5.50025-6
65. Rall, D., Menne, D., Schweidtmann, A.M., Kamp, J., von Kolzenberg, L., Mitsos, A., Wessling, M.: Rational design of ion separation membranes. J. Membr. Sci. **569**, 209–219 (2019). https://doi.org/10.1016/j.memsci.2018.10.013
66. Rasmussen, C.E.: Gaussian processes in machine learning. In: Advanced lectures on machine learning, pp. 63–71. Springer (2004)
67. Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Comput. Chem. Eng. **19**(5), 551–566 (1995)
68. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. (1989). https://doi.org/10.1214/ss/1177012413

69. Schweidtmann, A.M., Clayton, A.D., Holmes, N., Bradford, E., Bourne, R.A., Lapkin, A.A.: Machine learning meets continuous flow chemistry: Automated optimization towards the pareto front of multiple objectives. Chem. Eng. J. (2018). https://doi.org/10.1016/j.cej.2018.07.031
70. Schweidtmann, A.M., Mitsos, A.: Deterministic global optimization with artificial neural networks embedded. J. Optim. Theory Appl. **180**(3), 925–948 (2019). https://doi.org/10.1007/s10957-018-1396-0
71. Schweidtmann, A.M., Netze, L., Mitsos, A.: Melon: Machine learning models for optimization. https://git.rwth-aachen.de/avt.svt/public/MeLOn/ (2020)
72. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: a review of Bayesian optimization. Proc. IEEE **104**(1), 148–175 (2016). https://doi.org/10.1109/JPROC.2015.2494218
73. Smith, E.M., Pantelides, C.C.: Global optimisation of nonconvex MINLPs. Comput. Chem. Eng. **21**, S791–S796 (1997). https://doi.org/10.1016/S0098-1354(97)87599-0
74. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp. 2951–2959 (2012)
75. Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.: Gaussian process optimization in the bandit setting: no regret and experimental design. arXiv preprint arXiv:0912.3995 (2009)
76. Stuber, M.D., Scott, J.K., Barton, P.I.: Convex and concave relaxations of implicit functions. Optim. Methods Softw. **30**(3), 424–460 (2015). https://doi.org/10.1080/10556788.2014.924514
77. Sundararajan, S., Keerthi, S.S.: Predictive approaches for choosing hyperparameters in Gaussian processes. In: Advances in Neural Information Processing Systems, pp. 631–637 (2000)
78. Tardella, F.: On the existence of polyhedral convex envelopes. In: Floudas, C.A., Pardalos, P. (eds.) Frontiers in Global Optimization, pp. 563–573. Kluwer Academic Publishers, Dordrecht (2004)
79. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005). https://doi.org/10.1007/s10107-005-0581-8
80. Tawarmalani, M., Sahinidis, N.V., Pardalos, P.: Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications. In: Nonconvex Optimization and Its Applications, vol. 65. Springer, Boston, MA (2002). https://doi.org/10.1007/978-1-4757-3532-1
81. Tsoukalas, A., Mitsos, A.: Multivariate McCormick relaxations. J. Glob. Optim. **59**(2–3), 633–662 (2014). https://doi.org/10.1007/s10898-014-0176-0
82. Ulmasov, D., Baroukh, C., Chachuat, B., Deisenroth, M.P., Misener, R.: Bayesian optimization with dimension scheduling: application to biological systems. In: Computer Aided Chemical Engineering, vol. 38, pp. 1051–1056. Elsevier (2016). https://doi.org/10.1016/B978-0-444-63428-3.50180-6
83. Von Stosch, M., Oliveira, R., Peres, J., de Azevedo, S.F.: Hybrid semi-parametric modeling in process systems engineering: past, present and future. Comput. Chem. Eng. **60**, 86–101 (2014). https://doi.org/10.1016/j.compchemeng.2013.08.008
84. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
85. Wang, J., Hertzmann, A., Fleet, D.J.: Gaussian process dynamical models. In: Advances in Neural Information Processing Systems, pp. 1441–1448 (2006)
86. Wechsung, A., Scott, J.K., Watson, H.A.J., Barton, P.I.: Reverse propagation of McCormick relaxations. J. Glob. Optim. **63**(1), 1–36 (2015). https://doi.org/10.1007/s10898-015-0303-6
87. Wiebe, J., Cecílio, I., Dunlop, J., Misener, R.: A robust approach to warped Gaussian process-constrained optimization. arXiv:2006.08222 (2020)
88. Wilson, J., Hutter, F., Deisenroth, M.: Maximizing acquisition functions for Bayesian optimization. In: Advances in Neural Information Processing Systems, pp. 9884–9895 (2018)

Affiliations

Artur M. Schweidtmann[1] · Dominik Bongartz[1] · Daniel Grothe[1] · Tim Kerkenhoff[1] · Xiaopeng Lin[1] · Jaromił Najman[1] · Alexander Mitsos[1,2,3]

✉ Alexander Mitsos
amitsos@alum.mit.edu

[1] Process Systems Engineering (AVT.SVT), RWTH Aachen University, Aachen, Germany

[2] JARA-CSD, 52056 Aachen, Germany

[3] Institute of Energy and Climate Research, Energy Systems Engineering (IEK-10), Forschungszentrum Jülich GmbH, 52425 Jülich, Germany