

On the development of a practical Bayesian optimization algorithm for expensive experiments and simulations with changing environmental conditions

Mike Diessner¹ , Kevin J. Wilson² and Richard D. Whalley³

¹School of Computing, Newcastle University, Urban Science Building, Newcastle upon Tyne, United Kingdom

²School of Mathematics, Statistics and Physics, Newcastle University, Herschel Building, Newcastle upon Tyne, United Kingdom

³School of Engineering, Newcastle University, Stephenson Building, Newcastle upon Tyne, United Kingdom

Corresponding authors: Mike Diessner and Richard D. Whalley; Emails: m.diessner2@newcastle.ac.uk; richard.whalley@newcastle.ac.uk

Received: 05 February 2024; **Revised:** 05 June 2024; **Accepted:** 15 September 2024



Keywords: Bayesian optimization; black-box optimization; computer emulator; Gaussian processes; wind farm optimization

Abstract

Experiments in engineering are typically conducted in controlled environments where parameters can be set to any desired value. This assumes that the same applies in a real-world setting, which is often incorrect as many experiments are influenced by uncontrollable environmental conditions such as temperature, humidity, and wind speed. When optimizing such experiments, the focus should be on finding optimal values conditionally on these uncontrollable variables. This article extends Bayesian optimization to the optimization of systems in changing environments that include controllable and uncontrollable parameters. The extension fits a global surrogate model over all controllable and environmental variables but optimizes only the controllable parameters conditional on measurements of the uncontrollable variables. The method is validated on two synthetic test functions, and the effects of the noise level, the number of environmental parameters, the parameter fluctuation, the variability of the uncontrollable parameters, and the effective domain size are investigated. ENVBO, the proposed algorithm from this investigation, is applied to a wind farm simulator with eight controllable and one environmental parameter. ENVBO finds solutions for the entire domain of the environmental variable that outperform results from optimization algorithms that only focus on a fixed environmental value in all but one case while using a fraction of their evaluation budget. This makes the proposed approach very sample-efficient and cost-effective. An off-the-shelf open-source version of ENVBO is available via the NUBO Python package.

Impact Statement

Providing a practical method for the optimization of expensive experiments and simulations under changing and externally given environmental conditions allows solving engineering problems that better mimic the challenges found in real-world applications. It makes results robust when transferring them from the lab to the real world as it removes simplistic assumptions that fail to hold in realistic scenarios. Instead of running one experiment multiple times under different environmental conditions, ENVBO, the proposed method, requires only a single optimization run that shares all available information about the controllable variables and the environmental conditions. This decreases the number of observations needed and lowers the cost of optimization, which is the main challenge of expensive black-box optimization.

  This research article was awarded Open Data and Open Materials badges for transparent practices. See the Data Availability Statement for details.

© The Author(s), 2024. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.



1. Introduction

Bayesian optimization is a sample-efficient optimization algorithm for the optimization of expensive-to-evaluate functions that do not possess a mathematical expression or where the expression is too complex to be solved analytically (Moćkus 1974, 1989; Žilinskis 1975; Jones et al. 1998; Snoek et al. 2012). Examples of these functions are physical experiments and computer simulations. Optimization means finding the optimal parameter values that maximize some objective. In its original form, Bayesian optimization is a global optimization algorithm that aims to find a global optimum of a function in a minimum number of function evaluations, also called observations. For Bayesian optimization to be effective, all parameters must be controllable, and all environmental factors influencing the output must remain constant. However, this assumption is only true in completely isolated and controlled environments. Considering more realistic scenarios or experiments where some variables cannot be controlled, it is questionable if this assumption holds. Environments in the real world are generally more complex, and environmental conditions, such as humidity, temperature, and wind speed, are typically given by uncontrollable external factors.

Many applications of Bayesian optimization—implicitly or explicitly—assume a simplistic world where all environmental conditions are fixed. In active flow control, for example, where the goal is to control blowing actuators to maximize the reduction of the skin friction drag over a flat plate, the ambient wind speed is assumed to be fixed (Mahfoze et al. 2019; Diessner et al. 2022; O'Connor et al. 2023; Mallor et al. 2024). However, the optimal parameters found from these simulations and experiments give optima for specific wind speeds and cannot necessarily be generalized to other wind speeds. This approach requires repeating the experiment for each wind speed to ensure optimality. Because wind speeds are assumed to be fixed, it is impossible to share observations and, thus, information between experiments. While observations from different wind speeds will likely not result in the same drag reduction, they will be correlated and contain some information that can be transferred to problems with varying wind speeds. Sharing information between different environmental conditions could decrease the number of necessary observations and make Bayesian optimization more sample-efficient and cost-effective—both essential properties and the main objectives of Bayesian optimization.

This article presents a practical strategy for optimizing expensive black-box functions such as physical experiments and computer simulations, with influential environmental conditions that are given by external circumstances and cannot be controlled during the optimization. The strategy extends Bayesian optimization by (a) fitting a global surrogate model over all controllable and uncontrollable variables, (b) solving the acquisition function conditionally on measurements taken for the uncontrollable variables, and (c) restricting the initial training data, which typically consists of many observations generated via a space-filling design to only one observation. It is shown that ENVBO, the proposed algorithm, generalizes to situations with noisy observations, multiple uncontrollable variables, and uncontrollable variables with different levels of fluctuation and variability.

To illustrate the value of this approach to the field of engineering, a wind farm simulator is considered to maximize the annual power generation by finding optimal positions for four wind turbines. The wind direction affects the power generation significantly and is assumed to vary randomly in these simulations. Thus, the wind direction represents an influential environmental condition. Results show that ENVBO outperforms two other optimization algorithms used as benchmarks in all but one case. It has the additional benefits of using fewer function evaluations than the benchmarks and can predict wind turbine positions for any possible wind direction within the investigated range. Similar results from the benchmarks could only be achieved by repeating simulation campaigns many times for different wind speeds—an expensive, if not infeasible, task. Thus, the proposed algorithm is sample-efficient and cost-effective and effectively addresses the main problem of expensive black-box function optimization. An off-the-shelf open-source version of ENVBO is available via the NUBO Python package (Diessner et al. 2023) at www.nubopy.com.

This research article is structured as follows. [Section 2](#) gives an overview of related literature and highlights differences to the work in this article. [Section 3](#) introduces Bayesian optimization, including

surrogate modeling and acquisition functions, and extends it to allow optimization with changing environmental conditions. [Section 4](#) validates the method on two synthetic test functions—the two-dimensional Levy function and the six-dimensional Hartmann function—and introduces a way of simulating randomly changing environmental conditions via random walks. [Section 5](#) investigates the five properties of the proposed method: noise, number of uncontrollable variables, parameter fluctuation, parameter variability, and effective domain size (i.e., the actual searched space given by the environmental conditions). [Section 6](#) considers a nine-dimensional wind farm simulator with eight controllable and one uncontrollable variable. [Section 7](#) discusses the results of the empirical investigation and the application to the simple wind farm simulator and highlights limitations and implications. Finally, a conclusion is drawn in [Section 8](#).

2. Related work

The methodology in this article is in the area of Bayesian optimization, whose origin can be traced back to the early 1960s when Kushner (1962) and Kushner (1964) introduced an optimization algorithm on a one-dimensional noisy function using a Gaussian process as a surrogate model and sampling heuristics that later evolved into the upper confidence bound (UCB) and probability of improvement acquisition functions. Research from 1971 onward linked Bayesian optimization predominantly to the optimization of expensive multimodal functions and introduced new heuristics—notably expected improvement (EI; Šaltenis 1971) and knowledge gradient (Moćkus 1972, 1974, 1989). Limitations in computational capabilities at the time resulted in using cheaper models, such as Wiener and Ornstein–Uhlenbeck processes, for which both heuristics collapse onto each other (Garnett 2023). The evolution of technology allowed computationally expensive models such as Gaussian processes to be used in the late 1990s, which incentivized reevaluation of the methods proposed two decades earlier. Schonlau (1997) and Jones et al. (1998) introduced their efficient global optimization (EGO) algorithm and studied EI in a deterministic setting. Auer (2003) and Srinivas et al. (2010) reexamined UCB and studied its theoretical regret for the first time, while Frazier and Powell (2007) and Scott et al. (2011) studied the knowledge gradient for discrete and continuous input spaces, respectively. Villemonteix et al. (2006) introduced an information-based strategy into the Bayesian optimization framework that was later studied and further developed into entropy search (Hennig and Schuler 2011), predictive entropy search (Hernández-Lobato et al. 2014), and max-value entropy search (Wang and Jegelka 2017). Comprehensive surveys are available from Brochu et al. (2010), Shahriari et al. (2015) and Frazier (2018). Closely related to Bayesian optimization is the work on multiarmed bandits investigating a similar problem but in a discrete parameter space (Garnett 2023; Berry and Fristedt 1985; Lattimore and Szepesvári 2020).

Bayesian optimization has been applied to many scientific fields, predominantly in the science, technology, engineering, and mathematics areas. Applications include fields such as material science (Frazier and Wang 2015; Packwood 2017) and drug discovery (Negoescu et al. 2011) in chemistry, physics (Duris et al. 2019), and biology (Li et al. 2018). In engineering, it has been used in civil engineering (Gramacy et al. 2016; Garnett et al. 2010), electrical engineering (Lyu et al. 2018), mechanical engineering (Sterling et al. 2015), and robotics (Martinez-Cantin et al. 2009; Calandra et al. 2015). Furthermore, Bayesian optimization is often used in machine learning and artificial intelligence, for example, to configure algorithms (Hutter et al. 2011) and to tune hyperparameters (Snoek et al. 2012). Garnett (2023) provides a thorough list of applications.

This article focuses on a particular case of Bayesian optimization (Frazier 2018) that is referred to as random environmental conditions (Chang et al. 2001), multitask optimization (Swersky et al. 2013), and contextual optimization (Krause and Ong 2011). Methods differ by the type of input parameters (discrete or continuous), the number of allowed contexts (finite or infinite), and the overall goal of the optimization (one optimal solution for all contexts or one optimal solution for each context). When the number of contexts is infinite, finding one optimal solution for each context corresponds to finding a function that returns optimal inputs based on the context. This is the main objective of this article. Gaussian process-based methods aiming to optimize problems with different contexts—that is, with controllable and environmental variables—considered in the past can be classified primarily into two types based on their optimization goal.

The first type aims to find one solution that yields the best result for all tasks/contexts. It is assumed that environmental variables take values according to a probability distribution. For example, if the environment is defined by its temperature, the current temperature will follow a distribution. Optimization aims to find the optimizer that maximizes the objective function, taking into account the likelihood of the different environments. Different approaches assume the distribution of the environmental variables to be discrete (Williams et al. 2000; Swersky et al. 2013), continuous (Groot et al. 2010; Xie et al. 2012), or both (Toscano-Palmerin and Frazier 2018). These methods assume that parameters and contexts can be selected during optimization, and many use a sequential approach, where parameters for the next candidate are selected before the context is chosen. Only Toscano-Palmerin and Frazier (2018) present a method that chooses parameters and contexts jointly and can optimize problems where contexts are uncontrollable and given randomly. Chang et al. (1999) and Chang et al. (2001) use this type of method to design a femoral component for hip replacements conditional on joint force orientation and cancellous bone properties. The aim is to find one optimal design for a wide demographic with varying characteristics.

The second type of method aims to find one solution for each context. Thus, interest lies not in finding one global optimum but multiple optima, one for each combination of environmental conditions. Pearce and Branke (2017), Char et al. (2019), and Chung et al. (2020) consider multiple discrete tasks, while Ginsbourger et al. (2014) and Pearce and Branke (2018) consider continuous environmental variables. These methods are closely related to the objective of this article. However, they have one important distinction. While the environmental values are given externally in real-world applications, it is assumed that they can be set to any desired values in the experiments and simulations above. This deviates from our problem formulation, where we explicitly regard problems with uncontrollable environmental variables—in experiments, simulations, and the real world.

The research in this article is closely related to Krause and Ong (2011), who modified UCB (Srinivas et al. 2010) to be suited to optimization with externally given environmental conditions and derived theoretical bounds for its contextual regret. In contrast to Krause and Ong (2011), this article considers improvement-based acquisition functions, that is, EI (Jones et al. 1998) and log EI (LogEI; Ament et al. 2023), gives a detailed description of the practical implementation of Bayesian optimization with environmental conditions and provides all code at <https://github.com/mikedieessner/environmental-conditions-BO>. In addition, the reported approach makes fewer assumptions than Krause and Ong (2011), who focus on a linear and additive covariance structure for the environmental variables. These assumptions can be restrictive as no information about the objective function might be available—as is often the case with expensive, derivative-free black-box optimization. Thus, a widely applicable strategy is required that this article aims to provide.

The area of multitask optimization is also related to multi-objective optimization. In multi-objective optimization, the aim is no longer to optimize one single objective function but rather multiple objective functions simultaneously. A global optimum for all objective functions is only possible if it happens to be at the same values for all inputs for all functions. Thus, the main goal is to find the best trade-off between the objective functions (Garnett 2023; Frazier 2018). This is related to the first problem above, which aims to find a trade-off between all contexts. However, objective functions in multi-objective optimization are generally assumed not to be correlated, while correlations are leveraged for contexts. Acquisition functions such as EI (Emmerich et al. 2006; Ponweiser et al. 2008; Yang et al. 2019b; Yang et al. 2019a), predictive entropy search (Hernández-Lobato et al. 2015) and max-value entropy search (Belakaria et al. 2020; Fernández-Sánchez et al. 2023) have been extended to allow multi-objective optimization.

3. Methodology

This section provides a brief overview of the fundamentals of Bayesian optimization, particularly the surrogate modeling via Gaussian processes and the acquisition functions used to guide the optimization. It

then introduces the novel conditional Bayesian optimization algorithm that enables optimization with uncontrollable environmental variables.

3.1. Bayesian optimization

Consider the d -dimensional maximization problem

$$\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}), \quad (1)$$

where \mathcal{X} is a continuous input space that is bounded by a hyper-rectangle such that $\mathcal{X} \in [a, b]^d$ with $a, b \in \mathbb{R}$. The objective function $f(\mathbf{x})$ typically has three properties. First, it is a black-box function that can be provided with an input vector \mathbf{x}_i and allows the observation of the scalar output y_i . Beyond this, no other information can be inferred from the function. Second, the function is expensive to evaluate, and significant time, resources, and money costs are generated at each evaluation. Third, the function typically lacks a derivative or is too expensive to compute (Frazier 2018). Any noise ϵ introduced during the evaluation of the objective function $f(\mathbf{x})$ is assumed to be independent and identically distributed Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ such that an observation can be defined as $y_i = f(\mathbf{x}_i) + \epsilon$. Multiple observation pairs consisting of inputs and outputs are defined as $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. In this article, $\mathbf{X}_n = \{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{y}_n = \{y_i\}_{i=1}^n$ are used to describe all training inputs and their corresponding outputs.

Bayesian optimization (Moćkus 1974, 1989; Žilinskas 1975; Jones et al. 1998; Srinivas et al. 2010; Snoek et al. 2012; Shahriari et al. 2015; Frazier 2018; Gramacy 2020) is an optimization algorithm based on surrogate modeling to solve expensive problem (1) in a minimum number of function evaluations. The expensive and opaque nature of these problems requires a sample-efficient optimization algorithm that keeps costs low to make the optimization feasible. Bayesian optimization has emerged as a prime candidate using a surrogate model \mathcal{M} to represent the unknown objective function $f(\mathbf{x})$. The mean of the surrogate model's predictive distribution is then used to compute an acquisition criterion $\alpha(\cdot)$ that guides the optimization process by proposing new input points to be evaluated by the objective function. Bayesian optimization is a sequential optimization algorithm performed in a feedback loop, as illustrated in Algorithm 1. This loop consists of three steps. First, the surrogate model is fitted to the training data \mathcal{D}_n . Second, the next candidate point \mathbf{x}_{n+1} is computed by maximizing the acquisition criterion α . Third, the objective function evaluates the new candidate point, and its output y_{n+1} is observed. The loop then starts again, adding this newly observed candidate point to the training set. Thus, Bayesian optimization gathers more information sequentially with each loop. The algorithm stops when a predefined evaluation budget N is exhausted and returns the data pair with the highest observation as its solution.

Algorithm 1 Basic Bayesian optimization algorithm.

Require: Evaluation budget N , number of initial points n_0 , surrogate model \mathcal{M} , and acquisition function α .

Sample n_0 initial training data points \mathbf{X}_0 via a space-filling design (McKay et al. 1979) and gather observations \mathbf{y}_0 .

Set $n = 0$.

while $n \leq N - n_0$ **do.**

Fit surrogate model \mathcal{M} to training data $\mathcal{D}_n = \{\mathbf{X}_n, \mathbf{y}_n\}$.

Find \mathbf{x}_{n+1} that maximizes an acquisition criterion α based on model \mathcal{M} , that is, solve $\operatorname{argmax}_{\mathbf{x}} \alpha(\mathbf{x})$.

Evaluate \mathbf{x}_{n+1} , observing y_{n+1} .

Increment n .

end while.

return Point \mathbf{x}^* with the highest observation y^* .

Algorithm 1 is illustrated in Figure 1. The objective function (dashed line) is a simple one-dimensional function with one local optimum and one global optimum at $x = 8$. The algorithm is initialized with three observations (dark blue dots), and the surrogate model is fitted, providing its prediction (red line) and the corresponding uncertainty (blue area) in the form of 95% confidence intervals around the prediction. This model is used to compute the acquisition criterion (orange area)—in this case, EI (see Section 3.1.2)—that, when maximized, provides the next candidate point (dashed red line) to be evaluated from the objective function. Bayesian optimization is run for eight iterations, and the surrogate model is updated with each newly evaluated candidate point until the algorithm finds the optimum at iteration six.

3.1.1. Gaussian process

Different methods, such as random forest regression (Hutter et al. 2011; Lindauer et al. 2022), density ratio estimation (Bergstra et al. 2011; Tiao et al. 2021), and neural networks (Snoek et al. 2015; Springenberg et al. 2016; Maraval et al. 2022) have been considered for surrogate modeling in Bayesian optimization. However, Gaussian process regression (Rasmussen and Williams 2006; Gramacy 2020) has seen the widest adaption for the surrogate model \mathcal{M} and has been researched extensively since the 1970s (Brochu et al. 2010). The four main reasons for this adaption are: first, Gaussian processes are a flexible modeling strategy, capable of representing a wide range of possible continuous objective functions $f(\mathbf{x})$ (Brochu et al. 2010; Frazier 2018; Gramacy 2020; Garnett 2023). Second, they allow incorporating prior knowledge about the objective function by choosing the prior mean function and the prior covariance kernel (Brochu et al. 2010; Frazier 2018; Gramacy 2020). The latter is particularly important as it controls the smoothness of the possible representations of the objective function. Third, they return a prediction with corresponding uncertainty, both required for computing acquisition functions (Garnett 2023; Gramacy 2020). Fourth, they are appropriate for continuous objective functions, which are the main focus of Bayesian optimization, and work well with the limited amounts of data available in expensive black-box optimization (see Section 3.1). Due to these advantages, Gaussian processes are typically used in Bayesian optimization; thus, this article considers Bayesian optimization based on Gaussian process regression. Rasmussen and Williams (2006) provide a comprehensive overview of Gaussian process regression and its implementation.

A Gaussian process is a nonparametric regression model that returns a prediction and its corresponding uncertainty for an unobserved point and can be described as a distribution of possible objective functions. Mathematically, it is a finite collection of random variables that have a joint Gaussian distribution. A Gaussian process only requires a prior mean function $\mu_0(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}$ and a prior covariance kernel $\Sigma_0(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ to be fully defined. The mean vector $m(\mathbf{X}_n) := \mu_0(\mathbf{X}_n)$ and the $n \times n$ covariance matrix $K(\mathbf{X}_n, \mathbf{X}_n) := \Sigma_0(\mathbf{X}_n, \mathbf{X}_n)$ specify the multivariate normal distribution, also called the prior distribution, as

$$f(\mathbf{X}_n) \sim \mathcal{N}(m(\mathbf{X}_n), K(\mathbf{X}_n, \mathbf{X}_n)). \quad (2)$$

This study follows Snoek et al. (2012) and chooses the constant mean function given in Equation (3) as the prior mean function $\mu_0(\cdot)$ and the Matérn 5/2 kernel presented in Equation (4) as the prior covariance function $\Sigma_0(\cdot, \cdot)$. Snoek et al. (2012) argue that the Matérn 5/2 kernel is better suited for the optimization of realistic black-box problems than the often-used radial basis function kernel (Equation (5)) because the latter assumes the objective function to be very smooth. This assumption might be too strong for many problems, making the Matérn kernel a good alternative. However, the main advantage of using Gaussian processes as the surrogate model is the ability to incorporate prior knowledge about the objective function. Thus, the choice of the covariance kernel should always be informed by the available information about the given problem. For cases where information about the smoothness of the objective function is unavailable—which might be common for black-box problems—, Snoek et al. (2012) make a case in favor of the Matérn 5/2 kernel over the radial basis function kernel as a default.

The Matérn 5/2 kernel uses the distance between inputs $r = |\mathbf{x} - \mathbf{x}'|$ to compute the uncertainty around its prediction. The kernel uses the output scale σ_f^2 to scale the covariance, where smaller values correspond

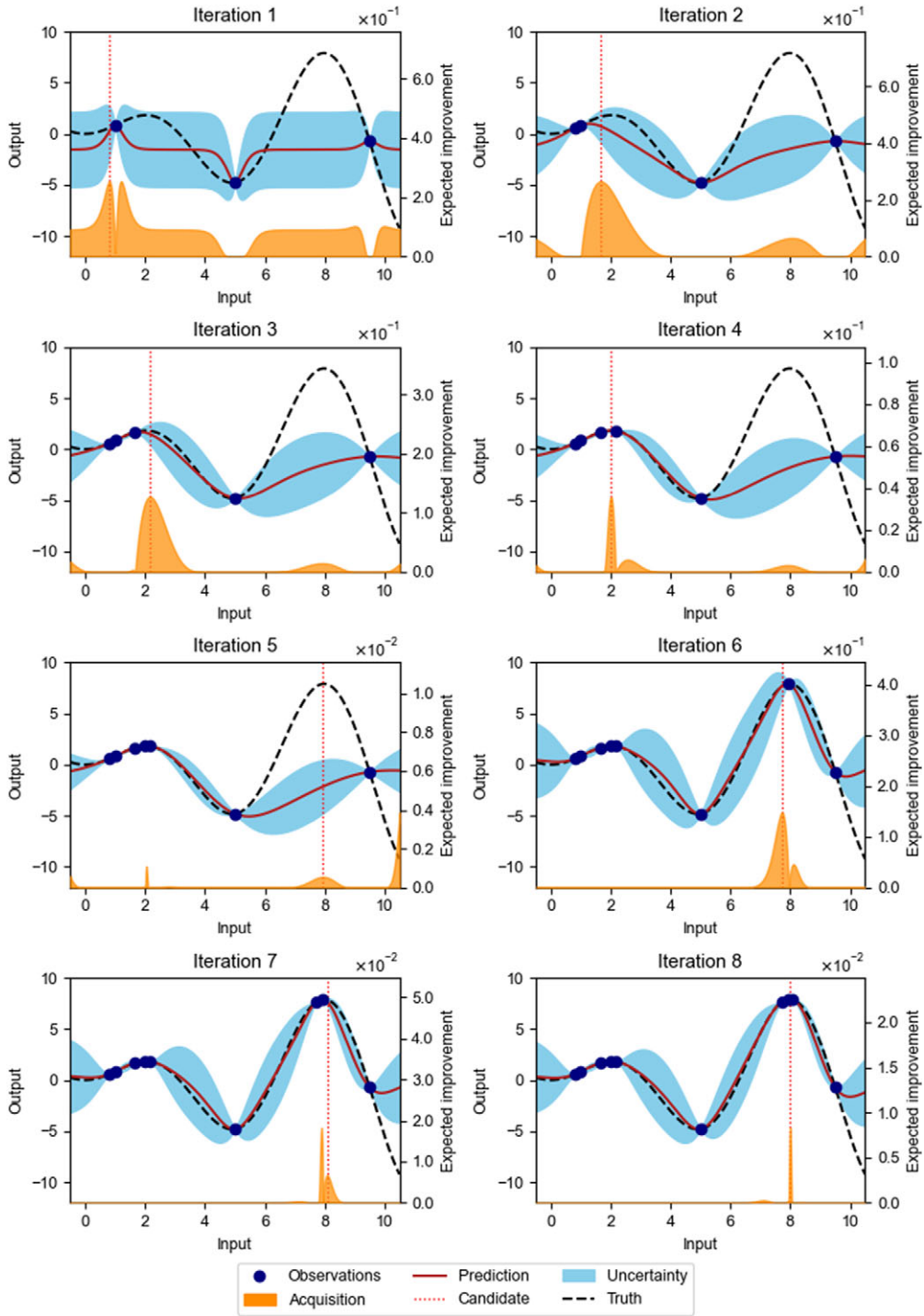


Figure 1. Bayesian optimization is applied to a one-dimensional function with one local and one global maximum. EI is used as the acquisition function. The input space is bounded by $[0, 10]$.

to a smaller deviation from its mean. The characteristic length scale l quantifies the extent to which outputs are correlated when moving along an input axis. Smaller length scales l mean shorter correlation lengths and more variable functions, while larger length scales l correspond to longer correlation lengths and more constant functions (Gramacy 2020; Rasmussen and Williams 2006).

$$\mu(\mathbf{x}) = c \tag{3}$$

$$\Sigma_{\text{Matérn}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2} \right) \exp\left(-\frac{\sqrt{5}r}{l} \right) \tag{4}$$

The radial basis function kernel given in Equation (5) is another popular alternative for the covariance function. It is much smoother than the Matérn kernel and thus is only suited for Bayesian optimization when it can be assumed that the underlying objective function is necessarily smooth (Snoek et al. 2012).

$$\Sigma_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{r^2}{2l^2} \right) \tag{5}$$

Covariance kernels can use one characteristic length scale l for all input dimensions d or d length scales $\mathbf{l} = \{l_i\}_{i=1}^d$, one for each input dimension d . While the former has less computational overhead due to only requiring one length scale for all dimensions, it is less flexible as it assigns the same correlation to each dimension. The latter considers each dimension individually and assigns an individual length scale to each dimension. This is known as automatic relevance determination (Neal 1996) as the Gaussian process assigns larger length scales to more constant dimensions, making them less influential in the computation of the covariance matrix. In contrast, inputs that are variable and change quickly are assigned smaller length scales, increasing their importance when computing the covariance kernel, as changes in these inputs generally affect the prediction significantly. Generally, the inverse of the length scales \mathbf{l} indicates the relevance of the corresponding input (Rasmussen and Williams 2006).

The Gaussian process described above has some hyperparameters θ that can be estimated from the training data by maximizing the log-marginal likelihood in Equation (6) via maximum likelihood estimation (Rasmussen and Williams 2006).

$$\begin{aligned} \log p(\mathbf{y}_n | \mathbf{X}_n) = & -\frac{1}{2} (\mathbf{y}_n - m(\mathbf{X}_n))^\top \left[K(\mathbf{X}_n, \mathbf{X}_n) + \sigma_y^2 \mathbf{I} \right]^{-1} (\mathbf{y}_n - m(\mathbf{X}_n)) \\ & -\frac{1}{2} \log |K(\mathbf{X}_n, \mathbf{X}_n) + \sigma_y^2 \mathbf{I}| - \frac{n}{2} \log 2\pi \end{aligned} \tag{6}$$

Besides the constant c in the mean function, the signal variance σ_f^2 and characteristic length scales \mathbf{l} in the covariance kernel, there is the noise variance σ_y^2 that reflects the noise level ε that is introduced independent of the objective function $f(\mathbf{x})$, such that $\theta = \{c, \sigma_f^2, \mathbf{l}, \sigma_y^2\}$.

The Gaussian process can be used to make predictions along with corresponding uncertainty quantification by computing the posterior predictive distribution (7). For n_* test points \mathbf{X}_* , it can be computed as the multivariate normal distribution conditional on the training data \mathcal{D}_n

$$f(\mathbf{X}_*) | \mathcal{D}_n, \mathbf{X}_* \sim \mathcal{N}(\mu_n(\mathbf{X}_*), \sigma_n^2(\mathbf{X}_*)) \tag{7}$$

$$\mu_n(\mathbf{X}_*) = K(\mathbf{X}_*, \mathbf{X}_n) \left[K(\mathbf{X}_n, \mathbf{X}_n) + \sigma_y^2 \mathbf{I} \right]^{-1} (\mathbf{y} - m(\mathbf{X}_n)) + m(\mathbf{X}_*) \tag{8}$$

$$\sigma_n^2(\mathbf{X}_*) = K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X}_n) \left[K(\mathbf{X}_n, \mathbf{X}_n) + \sigma_y^2 \mathbf{I} \right]^{-1} K(\mathbf{X}_n, \mathbf{X}_*), \tag{9}$$

are the covariance matrices of sizes $n_* \times n$, $n \times n$, and $n_* \times n_*$ between the training inputs \mathbf{X}_n and the test inputs \mathbf{X}_* , respectively.

3.1.2. Acquisition functions

Acquisition functions guide the sequential selection of candidate points by quantifying if a particular input point is likely to be a good new candidate point and thus should be evaluated by the objective function $f(\mathbf{x})$. This is achieved by computing and maximizing an acquisition criterion $\alpha(\cdot)$ based on the posterior distribution of the Gaussian process and the available training data \mathcal{D}_n . The criterion's exact form depends on the individual acquisition function; however, most acquisition functions have one property in common—the exploration-exploitation trade-off (Shahriari et al. 2015; Frazier 2018). Exploration can be defined as choosing candidate points from areas with high uncertainty, where no training data points were observed, and thus, little information is available. Conversely, exploitation is defined as selecting candidate points from areas with a high predictive mean, that is, points that are close to high training data points. To understand the importance of balancing exploration and exploitation, consider the extreme cases of pure exploration and pure exploitation. For the former, only points with the highest uncertainty would be selected. While this minimizes the uncertainty of the Gaussian process, it is not a sample-efficient approach as information about high predictive means is disregarded, and areas with the best-observed outputs are avoided. For the latter, the algorithm will blindly follow the best-performing points and never explore other areas. The algorithm will probably converge toward the first optimum it discovers, making it prone to getting stuck in a local optimum. Thus, a hybrid solution that combines exploration and exploitation is beneficial.

While related work (Krause and Ong 2011) focuses on a theoretical investigation of UCB (Srinivas et al. 2010), this study empirically investigates EI (Jones et al. 1998) and shows its superior performance on two test functions. One of the main advantages of EI over UCB is that it does not require the choice of a value for the trade-off parameter that balances exploration and exploitation. Finding optimal values for this parameter that perform well for specific problems is not trivial, although theoretical guarantees have been derived (Srinivas et al. 2010). EI has been investigated empirically (Jones et al. 1998; Jones 2001; Brochu et al. 2010), its convergence has been studied (Bull 2011), and it has shown efficiently in practice (Snoek et al. 2012). Although improvement-based methods are among the most popular and well-studied acquisition functions (Ament et al. 2023), other acquisition functions, such as knowledge gradient (Frazier et al. 2009; Frazier 2018) and information-based approaches, such as entropy search (Villemonteix et al. 2006; Hennig and Schuler 2011), predictive entropy search (Hernández-Lobato et al. 2015), and max-value entropy search (Wang and Jegelka 2017), are also available. While we focus on EI and UCB here, our method could be used with any choice of acquisition function.

EI (Jones et al. 1998) is an improvement-based acquisition function that aims to find candidates that perform better than a defined target, typically the best available training data point. EI is defined as

$$\alpha_{\text{EI}}(\mathbf{X}_*) = \begin{cases} (\mu_n(\mathbf{X}_*) - y^{\text{best}}) \Phi(z) + \sigma_n(\mathbf{X}_*) \phi(z) & \text{if } \sigma_n(\mathbf{X}_*) > 0 \\ 0 & \text{if } \sigma_n(\mathbf{X}_*) = 0 \end{cases}, \quad (10)$$

where $z = \frac{\mu_n(\mathbf{X}_*) - y^{\text{best}}}{\sigma_n(\mathbf{X}_*)}$, $\mu_n(\cdot)$ and $\sigma_n(\cdot)$ are the mean and the standard deviation of the Gaussian processes' posterior predictive distribution (7), y^{best} is the current best observation, and $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative distribution function and probability density function of the standard normal distribution $\mathcal{N}(0, 1)$.

Although EI can never be nonpositive mathematically, it can become zero numerically when computed due to the floating-point precision of the programming language. This results in flat areas where the EI is zero and cannot be optimized correctly. To prevent numerically vanishing values, LogEI was proposed by Ament et al. (2023) as

$$\alpha_{\text{LogEI}}(\mathbf{X}_*) = \begin{cases} \log_h \left(\frac{\mu_n(\mathbf{X}_*) - y^{\text{best}}}{\sigma_n(\mathbf{X}_*)} \right) + \log(\sigma_n(\mathbf{X}_*)) & \text{if } \sigma_n(\mathbf{X}_*) > 0 \\ 0 & \text{if } \sigma_n(\mathbf{X}_*) = 0 \end{cases}, \quad (11)$$

where

$$\log_h(z) = \begin{cases} \log(\phi(z) + z\Phi(z)) & \text{if } z > -1 \\ -z^2/2 - c_1 + \log \text{Imexp}(\log(\text{erfcx}(-z/\sqrt{2})|z|) + c_2) & \text{if } -1/\sqrt{\epsilon} < z \leq -1, \\ -z^2/2 - c_1 - 2\log(|z|) & \text{if } z \leq -1/\sqrt{\epsilon} \end{cases} \quad (12)$$

where $c_1 = \log(2\pi)/2$, $c_2 = \log(\pi/2)/2$, ϵ is the numerical precision, and $\log \text{Imexp}$ and erfcx are stable implementations of $\log(1 - \exp(z))$ and $\exp(z^2)\text{erfc}(z)$, respectively, where erfc is the complementary error function.

The UCB (Srinivas et al. 2010) is an optimistic acquisition function and assumes the uncertainty of the posterior Gaussian process to be true to a predefined level. It can be computed as

$$\alpha_{\text{UCB}}(\mathbf{X}_*) = \mu_n(\mathbf{X}_*) + \sqrt{\beta}\sigma_n(\mathbf{X}_*), \quad (13)$$

where β is a predefined trade-off parameter that can be set for each iteration of the Bayesian optimization algorithm, it can be kept constant for the full optimization campaign or be varied at each iteration (Srinivas et al. 2010). Srinivas et al. (2010) investigated some theoretical properties of β , while Diessner et al. (2022) investigated how different values for β affect the optimization. As the acquisition functions in this study are deterministic, they can be maximized with a deterministic optimizer, such as L-BFGS-B (Zhu et al. 1997).

3.2. Changing environmental conditions

The Bayesian optimization algorithm in Algorithm 1 assumes that all parameters influencing the output can be controlled. However, in many cases, when optimizing physical experiments, variables will be present that influence the output but cannot be controlled. This article refers to these uncontrollable variables as environmental variables as they are externally given within the ambient environment of the experiment. Examples of such uncontrollable variables are temperature, humidity and wind speed. This section presents an extension to Algorithm 1 that allows the inclusion of uncontrollable environmental variables in the optimization process. The extension can be broken down into three parts as highlighted in Algorithm 2.

The main modification to Algorithm 1 concerns the surrogate modeling. The basic Bayesian optimization algorithm fits a surrogate model over all controllable variables. Environmental variables are not included and are assumed to be fixed over the full optimization process or are irrelevant to the output. Algorithm 2 does not make this assumption and includes all controllable parameters \mathbf{x}_C and environmental variables \mathbf{x}_E in its surrogate model. The inputs \mathbf{X}_n of the training data $\mathcal{D}_n = \{\mathbf{X}_n, \mathbf{y}_n\}$ are extended from $\mathbf{X}_n = \{\mathbf{X}_{n,C}\}$ to $\mathbf{X}_n = \{\mathbf{X}_{n,E}, \mathbf{X}_{n,C}\}$.

The second extension regards the computation of the next candidate point, specifically, the maximization of the acquisition function. While in Algorithm 1 all parameters are assumed to be controllable and the acquisition function can be maximized over all parameters $\max_{\mathbf{x}_C} \alpha(\mathbf{x}_C)$, Algorithm 2 must differentiate between the controllable parameters \mathbf{x}_C and environmental variables \mathbf{x}_E . The uncontrollable variables are given by the environment and can only be measured but not manipulated. Hence, the maximization of the acquisition function is broken down into two steps. First, the environmental variables are measured. This gives values for the uncontrollable inputs for the next candidate point $\mathbf{x}_{n+1,E}$. Second, the acquisition function is maximized conditional on these values for the environmental inputs $\max_{\mathbf{x}_C|\mathbf{x}_E} \alpha(\mathbf{x}_C)$ resulting in the controllable inputs for the next candidate point $\mathbf{x}_{n+1,C}$. Conditional maximization essentially means that the environmental variables \mathbf{x}_E are treated as fixed for the maximization of the acquisition function for one iteration. This assumes that the environmental variables do not change significantly from the time of measuring until the evaluation of the new candidate point \mathbf{x}_{n+1} . This assumption should be realistic for most experiments, as one iteration of the Bayesian optimization loop—that is, measuring the environmental variables, fitting a Gaussian process and optimizing the acquisition function conditional on the measurements—takes only a few seconds (see Diessner et al. (2023) for runtimes of different Bayesian optimization packages). However, issues could arise when working with

environmental variables that change rapidly. Thus, Section 5 investigates the influence of different fluctuation rates. The new candidate point \mathbf{x}_{n+1} is then defined as a combination of the measurements for the environmental variables $\mathbf{x}_{n+1,E}$ and the results of the maximization of the acquisition function $\mathbf{x}_{n+1,C}$.

The last adjustment to Algorithm 1 focuses on generating the training data. Usually, training data are produced using a space-filling design, such as a Latin hypercube (McKay et al. 1979). Under the assumption that all parameters can be controlled, the experiment can be repeated for each training data point to observe its output. However, the modified Bayesian optimization algorithm includes uncontrollable variables in its computation. Thus, it is impossible to evaluate any arbitrary combination of inputs as it is limited by the current measurement of the environmental variables. To resolve this issue, Algorithm 2 uses one training data point \mathbf{x}_0 instead of multiple points generated from a space-filling design. The initial training data are restricted to a single point, while the second data point is already computed via Bayesian optimization. The first data point is generated by taking measurements for the environmental variables $\mathbf{x}_{0,E}$ and randomly selecting values for the controllable parameter $\mathbf{x}_{0,C}$. These inputs are then evaluated, resulting in a complete training inputs-output pair $\mathcal{D}_0 = \{\mathbf{x}_0, y_0\}$, followed by the first Bayesian optimization loop.

Algorithm 2 Modified Bayesian optimization algorithm with environmental conditions.

Require: Evaluation budget N , surrogate model \mathcal{M} , acquisition function α .

Sample an initial training data point $\mathbf{x}_0 = \{\mathbf{x}_{0,E}, \mathbf{x}_{0,C}\}$ where environmental parameters $\mathbf{x}_{0,E}$ are measured and controllable parameters $\mathbf{x}_{0,C}$ are randomly sampled and gather observation y_0 .

Set $n = 0$.

while $n \leq N - n_0$ **do**.

Fit surrogate model \mathcal{M} to training data $\mathcal{D}_n = \{\mathbf{X}_n, \mathbf{y}_n\}$, where $\mathbf{X}_n = \{\mathbf{X}_{n,E}, \mathbf{X}_{n,C}\}$.

Measure environmental variables $\mathbf{x}_{n+1,E}$.

Find values for the controllable parameters $\mathbf{x}_{n+1,C}$ that maximize the acquisition function α conditionally on the measurements $\mathbf{x}_{n+1,E}$ such that $\mathbf{x}_{n+1} = \{\mathbf{x}_{n+1,E}, \mathbf{x}_{n+1,C}\}$, that is, solve $\operatorname{argmax}_{\mathbf{x}_C | \mathbf{x}_E} \alpha(\mathbf{x}_C)$.

Evaluate \mathbf{x}_{n+1} by observing y_{n+1} .

Increment n .

end while.

return Point \mathbf{x}^* with the highest observation y^* .

In contrast to Krause and Ong (2011), the proposed approach does not assume different covariance structures for controllable and environmental variables. When working with experiments and simulators, the underlying objective function is generally unknown or too complex to compute directly. Even with expert knowledge, there might be insufficient information about these black boxes to confidently assume a linear or additive structure for the environmental variable (Frazier 2018). Hence, providing the surrogate model with enough flexibility to estimate the covariance structure is essential. This can be achieved using the Matérn kernel for controllable and environmental variables—or the radial basis function kernel for very smooth objective functions.

Figure 2 illustrates the conditional variable optimization on a two-dimensional problem with one uncontrollable variable x_1 and one controllable parameter x_2 . The upper-left plot shows the true output of the objective function, where yellow areas indicate high function values and blue areas indicate low function values. The goal is to find the optimal value for the controllable parameter (y -axis) that maximizes the output for any uncontrollable variable (x -axis) value. The upper-right plot shows the predictive mean of a Gaussian process fitted to 20 training data points (black crosses). Following Algorithm 2, a measurement (red dashed line) of the uncontrollable variable results in $x_1 = -0.5$. The next iteration of the optimization loop is performed conditional on this measurement. The lower-left plot

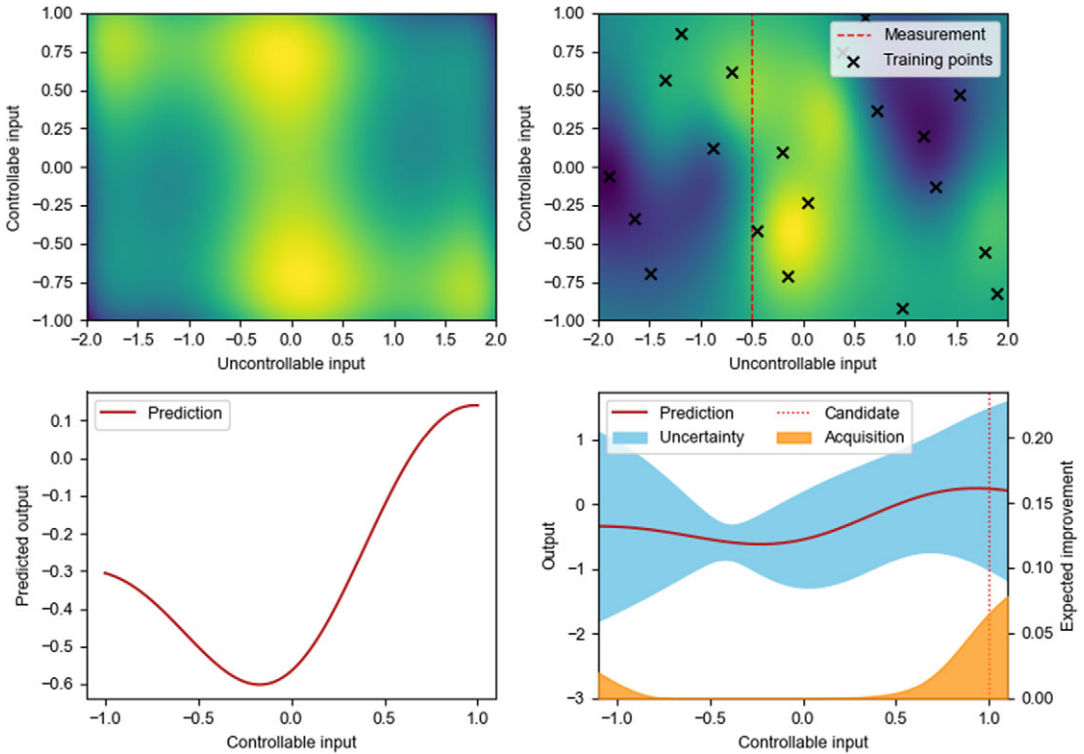


Figure 2. Maximization of a two-dimensional problem with one environmental variable x_1 and one controllable variable x_2 . Yellow areas indicate high outputs, and dark blue areas indicate low outputs. Upper-left: True objective function. Upper-right: Prediction of a Gaussian process with a measurement taken for the following conditional optimization step. Lower-left: Gaussian process prediction for optimization conditional on the measurement. Lower-right: Bayesian optimization step conditional on the measurement.

shows the predictive mean of the Gaussian process for $x_1 = -0.5$. The conditional optimization takes a slice from the full surrogate model. It reduces the two-dimensional optimization problem to a one-dimensional problem for each optimization step, where only the controllable parameters are considered. However, the information gained from the data is shared between each iteration. Notice that no training points lie on the measurement line; however, the model uses the available training points to inform its prediction. If Algorithm 1 were used, the uncontrollable input would be assumed to be fixed for the entire optimization loop, and the optimization process would need repeating for each value of x_1 . The lower-right plot extends the lower-left plot by adding the uncertainty from the Gaussian process and the acquisition function. The optimal value of the controllable input x_2 is found by maximizing the acquisition function, and the new candidate point is a combination of this maximum and the measurement taken for the uncontrollable variable x_1 . The candidate point is observed and added to the training data to be used in the next iteration of the optimization loop.

4. Simulations

This section introduces two synthetic test functions¹—the Levy function and the Hartmann function—and applies the Bayesian optimization algorithm with environmental conditions presented in Section 3.2.

¹ See <https://www.sfu.ca/ssurjano/optimization.html> for further details on the synthetic test functions.

Simulations for both problems are run for 100 evaluations and are repeated 30 times² to validate the robustness of Algorithm 2. This decreases the risk that results are influenced by the method's inherent randomness, for example, the randomly sampled training points that initialize the algorithm. Both problems assume one uncontrollable variable whose value is provided by a random walk at each iteration. In the simulations, each step of the random walk adds a sample from a uniform distribution \mathcal{U} to the previous value of the uncontrollable variable, such that

$$\mathbf{x}_{n,E} = \mathbf{x}_{n-1,E} + \mathcal{U}_{[-\mathbf{a},\mathbf{a}]}, \quad (14)$$

where \mathbf{a} is a vector of small predefined constants that provide the minimal and maximal change of the environmental variables from one iteration to the next. This uniform assumption represents the natural fluctuation of the uncontrollable variables encountered in a real-world application, for example, changes in temperature, humidity, and wind speed. It further allows the investigation of uncontrollable variables with different fluctuation levels by increasing or decreasing the constants in \mathbf{a} as discussed in Section 5. Assuming another distribution for the constants in \mathbf{a} , such as a Normal distribution, is an alternative to this approach.

Subsequent sections analyze the performance of Algorithm 2 by comparing predictions from the Gaussian process models $\mu_n(\mathbf{x})$ to the actual optimal values $f(\mathbf{x})$. In both cases, results are obtained by maximizing the Gaussian process prediction and the true objective function conditional on identical test values \mathbf{x}'_E for the uncontrollable variable. These test values are sampled from a maximin Latin hypercube design (McKay et al. 1979) within the observed domain of the uncontrollable variable $[\min(\mathbf{x}_{N,E}), \max(\mathbf{x}_{N,E})]$ considered by the algorithm. We also call this observed domain the effective domain. This ensures a fair comparison by avoiding predictions outside the effective domain requiring extrapolation. Extrapolation with Gaussian processes generally means that predictions default to the prior mean function. In cases where extrapolation cannot be avoided, making the prior mean function as informative as possible—for example, by going beyond zero and constant mean functions with polynomial and trigonometric mean functions—can improve results significantly (Planas et al. 2020). To score the performance of the algorithm, the mean absolute percentage error $\text{MAPE}(\mu_n(\mathbf{X}'_i), f(\mathbf{X}'_i)) = \frac{1}{m} \sum_{i=1}^m \left| \frac{\mu_n(\mathbf{x}'_i) - f(\mathbf{x}'_i)}{f(\mathbf{x}'_i)} \right|$ between the Gaussian process' predictions and the truths are computed for all m test points \mathbf{X}' .

The acquisition criterion conditional on values of the uncontrollable variables $\mathbf{x}_{n,E}$ is maximized with the SLSQP algorithm (Kraft 1994) using multiple starts. For this strategy, 100 points are sampled from a maximin Latin hypercube design (McKay et al. 1979) and evaluated by the acquisition criterion. The best 20 points are then used to initialize the SLSQP algorithm, and only the best result is used as the solution for the optimization problem. The multiple starts aim to reduce the risk of converging toward a local optimum instead of the desired global optimum of the acquisition function.

4.1. The two-dimensional Levy function

The Levy function

$$f(\mathbf{x}) = \sin^2(\pi w_1) + (w_1 - 1)^2 [1 + 10 \sin^2(\pi w_1 + 1)] + (w_2 - 1)^2 [1 + \sin^2(2\pi w_2)],$$

where $w_i = 1 + \frac{x_i - 1}{4}$, for $i = 1, 2$, is a two-dimensional function with two input parameters x_1 and x_2 . Although often used as a minimization problem to find the global minimum $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (1, 1)$ for the input space $[-10, 10]^2$, we choose the bounds of the parameters as $[-7.5, 7.5]$ and $[-10, 10]$, respectively to create a maximization problem that is better suited for testing Algorithm 2. The controllable parameter is restricted from its usual range of $[-10, 10]$ to $[-7.5, 7.5]$ to avoid the steep ridges of the Levy function at $x_1 = -10$ and $x_1 = 10$ that push the optima toward the outer bounds of the parameter space. The function given in Figure 3 shows a clear ridge at values of about $x_1 = -6$ for the controllable parameter x_2 for all

² Runs, replications, and repeats are used interchangeably in this article.

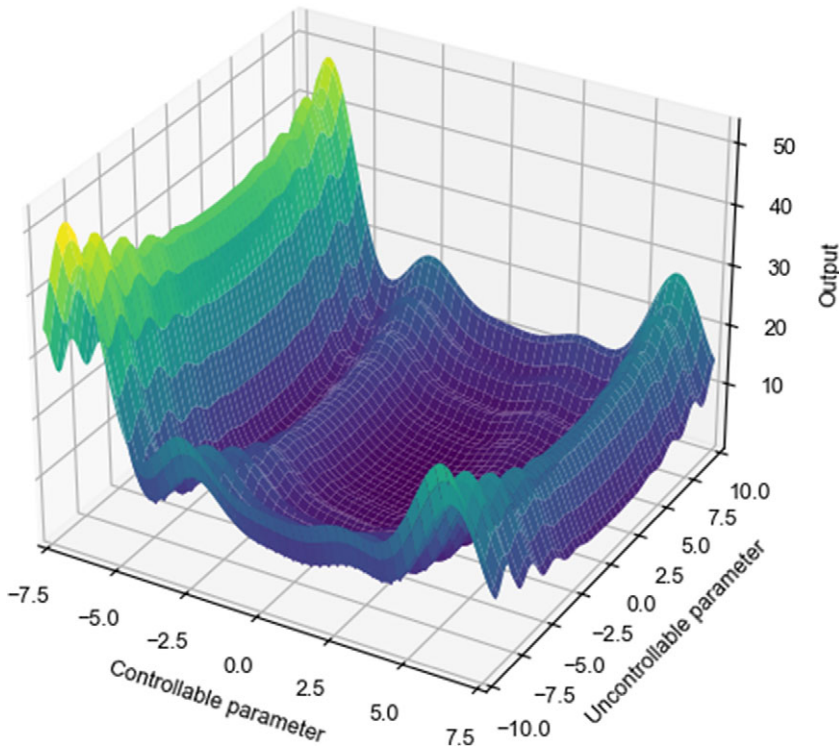


Figure 3. Two-dimensional negated Levy function with one controllable parameter x_1 bounded by $[-7.5, 7.5]$ and one uncontrollable variable x_2 bounded by $[-10, 10]$.

values of the uncontrollable variable x_1 . The simulations use $a = 1.5$ as the uniform distribution constant of the random walk for the uncontrollable variable x_2 . For this relatively simple two-dimensional function, the constant a was chosen to change by no more than 7.5% of the uncontrollable parameter's range in either direction per iteration. This assumes that the environmental conditions do not change too much between iterations. However, the effect of setting a to different values is investigated and discussed in Section 5.

The upper-left plot in Figure 4 shows the performance of Algorithm 2 as the mean absolute percentage error between the maximum of the Gaussian process prediction and the maximum of the true objective function conditional on 25 test values of the uncontrollable variable. Three alternatives for the acquisition function—EI, LogEI, and UCB introduced in Section 3.1.2—are compared to a benchmark where values for the controllable variable were selected randomly. Three different values (4, 8, 16) for the trade-off parameter β of UCB are considered in Figure 5. While results for all parameter values are comparable, UCB with trade-off parameter $\beta = 8$ performs slightly better in terms of mean performance and variance between runs. Thus, comparisons with EI and LogEI use $\beta = 8$. The solid lines indicate the mean performance, while the shaded areas indicate the 95% confidence interval over the 30 replications. Each replication's mean absolute percentage error is computed for every 10 evaluations for the same 25 test points \mathbf{X}'_E of the environmental variable. The test values are sampled from a Latin hypercube bounded by the minimal and maximal value of the uncontrollable variable after 100 function evaluations. The mean absolute percentage error starts just below 0.8 for all alternatives. While the improvement-based algorithms (EI and LogEI) performed better than the random benchmark, the algorithm using UCB performs worse. After 100 function evaluations, EI and LogEI have a mean absolute percentage error of 0.08 and 0.06, respectively, and improve upon the random benchmark (0.17). UCB only achieves a mean

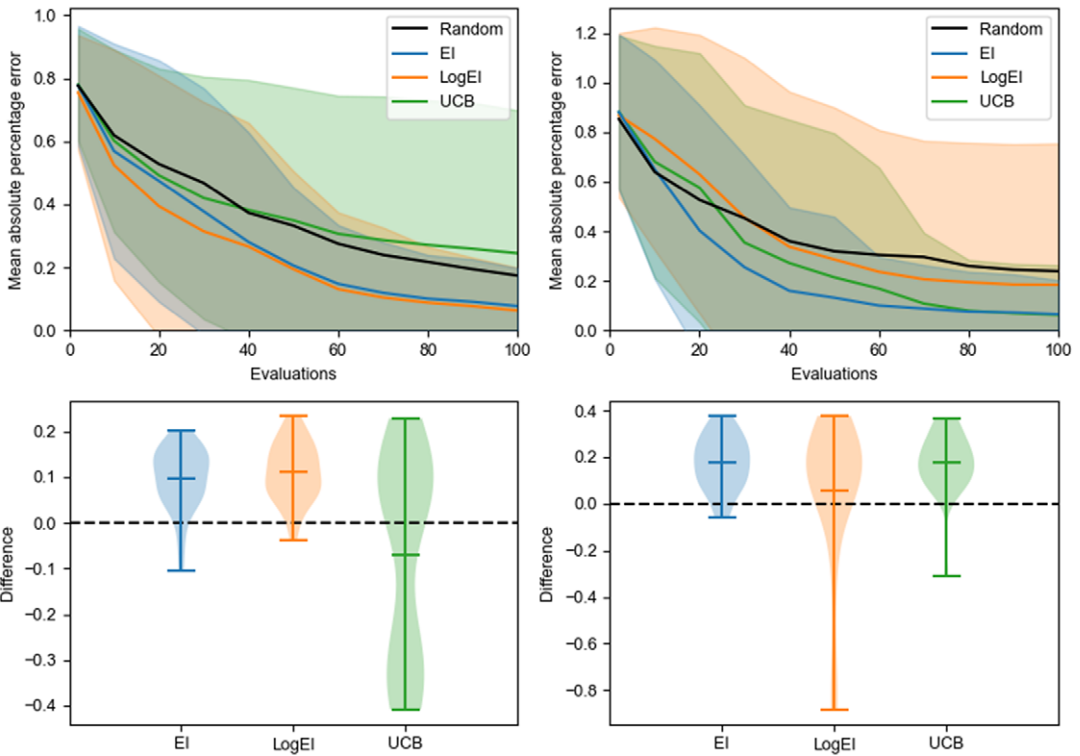


Figure 4. Upper row: Means (lines) and 95% confidence intervals (shaded areas) of the mean absolute percentage error between Gaussian process prediction and truth over 30 replications. Lower row: Difference between algorithms and random benchmark after 100 function evaluations for each of the 30 replications. Two-dimensional Levy function with one uncontrollable parameter on the left and six-dimensional Hartmann function with one uncontrollable parameter on the right.

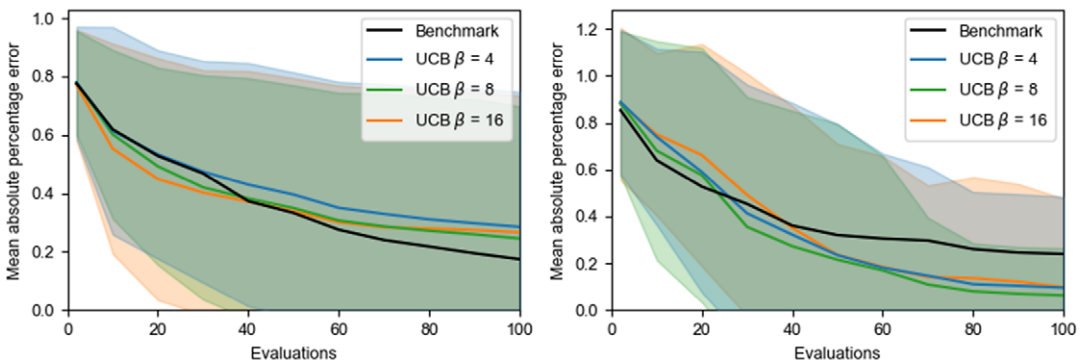


Figure 5. Comparison of different trade-off parameters β for the UCB acquisition function. Means (lines) and 95% confidence intervals (shaded areas) of the mean absolute percentage error between Gaussian process prediction and truth over 30 replications. Two-dimensional Levy function with one uncontrollable parameter on the left and six-dimensional Hartmann function with one uncontrollable parameter on the right.

absolute percentage error of 0.24. Moreover, the optimistic strategy shows large confidence intervals, indicating that the method is not robust. Altering the trade-off parameter β did not improve this result, as illustrated in the left plot in Figure 5.

The lower-left plot presents the difference between the mean absolute percentage error of the random benchmark and the three alternative acquisition functions after 100 evaluations. This difference is computed for each of the 30 replications, and distributions of the differences are plotted for the three different acquisition functions. Negative values indicate replications where the benchmark resulted in superior solutions, while positive values indicate that the given version of Algorithm 2 performed better than the benchmark. Although no alternative is better than the benchmark for every single replication of the 30 total replications, there is a clear difference between the improvement-based and the optimistic acquisition functions. Indeed, the mean of UCB (displayed by the horizontal line toward the center of each violin plot) is the only one worse than zero. This means that the random benchmark outperforms UCB on average. Additionally, the plot mirrors the lack of robustness discovered earlier by the large spread in differences. While the method performs better for some replications than the random benchmark, it performs much worse for others. A Mann–Whitney U test was performed to determine what alternatives perform differently from the random benchmark to a 1% significance level. The test returned a P -value of 0.0 for the improvement-based algorithms and 0.37 for the UCB. This rejects the hypothesis that methods perform equally well for the improvement-based methods, reinforcing the results from the visual analysis that EI and LogEI perform significantly better than the random benchmark. This cannot be said for the optimistic method—the test cannot reject the null hypothesis, indicating that UCB does not perform significantly differently from a random approach.

4.2. The six-dimensional Hartmann function

The negated³ Hartmann function

$$f(\mathbf{x}) = \sum_{i=1}^4 \alpha_i \exp\left(-\sum_{j=1}^6 \mathbf{A}_{ij}(x_j - \mathbf{P}_{ij})^2\right),$$

where

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T,$$

$$\mathbf{A} = \begin{pmatrix} 10.00 & 3.00 & 17.00 & 3.50 & 1.70 & 8.00 \\ 0.05 & 10.00 & 17.00 & 0.10 & 8.00 & 14.00 \\ 3.00 & 3.50 & 1.70 & 10.00 & 17.00 & 8.00 \\ 17.00 & 8.00 & 0.05 & 10.00 & 0.10 & 14.00 \end{pmatrix}, \text{ and}$$

$$\mathbf{P} = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix},$$

is a six-dimensional function with six input parameters $x_1, x_2, x_3, x_4, x_5,$ and x_6 that are evaluated on the hypercube $(0,1)^6$. It has six local maxima and one global maximum with $f(\mathbf{x}^*) = 3.32$ at $\mathbf{x}^* = (0.20, 0.15, 0.48, 0.28, 0.31, 0.66)$. The simulations use $a = 0.05$ as the uniform distribution constant of the random walk for the environmental variable x_6 . As this six-dimensional function is more complex than the Levy function, the constant a was chosen to change by a maximum of 5% of the uncontrollable

³Bayesian optimization problems are generally expressed as maximization problems as introduced in Section 3. The Hartmann function is a minimization problem and is negated for the following simulations to keep in line with this convention.

parameter's range in either direction, that is, the environmental conditions do not change too much between iterations. Section 5 sets a to 10%, 25%, 50%, and 100% and investigates their effect.

The upper-right plot in Figure 4 shows the performance of Algorithm 2 over the 30 repeats for the same three acquisition function as for the Levy function and compares it to the random benchmark. The mean absolute percentage error starts between 0.85 and 0.90 for all four algorithms. After 100 function evaluations, the mean absolute percentage error between the prediction of the Gaussian process and the true objective value for Algorithm 2 using EI and UCB with $\beta = 8$ are 0.07 and 0.06, respectively—much better than the random benchmark with 0.24. However, the algorithm using LogEI with a mean absolute percentage error of 0.18 after 100 evaluations performs only slightly better than the benchmark and significantly worse than the other versions of Algorithm 2. The large 95% confidence intervals for LogEI indicate that Algorithm 2 with LogEI is not robust in this case, making it less reliable than EI and UCB.

The violin plots of the difference between the mean absolute percentage error of the benchmark and the three variations of Algorithm 2 for all 30 replications on the lower-right in Figure 4 indicate that EI performs the best with almost all replications better than the random benchmark. LogEI, on the other hand, performs comparably to the benchmark on average but has a large spread, with some replication performing much worse. UCB performs similarly to EI but has an outlier that performs much worse than the random benchmark. Overall, EI is the most robust method and is not prone to outliers. Despite these differences between algorithms, results after 100 evaluations for all three algorithms are significantly different from the random results to a 1% significance level: the P -values from Mann–Whitney U tests are 0.0 for the EI and the UCB version of Algorithm 2 and 0.007 LogEI. While this shows that the results of all methods are significantly different from the benchmark, only EI and UCB perform better than the benchmark indicated by the better average mean absolute percentage error.

Figure 5 shows the results for different values of the trade-off parameter β . While the average performance is very similar after 100 evaluations, there is a difference in the 95% confidence intervals for the Hartmann function. However, no clear correlation is noticeable as $\beta = 8$ performs better than $\beta = 4$ and $\beta = 16$, indicating no clear trend.

5. Empirical analysis of properties

Based on the investigations of Section 4, we define ENVBO as a version of Algorithm 2 that uses the EI acquisition function. This section explores the effect of changes to five aspects of the underlying objective function or environmental conditions on ENVBO using the negated Hartmann function. The effect of adding different levels of random Gaussian noise to the function's output, the influence of more than one uncontrollable variable, the effect of the fluctuation level (i.e., the step size a of the random walk), the impact of the variability in the uncontrollable variable, and the relationship between the algorithm performance and the effective domain size of the uncontrollable variables are investigated. An off-the-shelf version of ENVBO is available via the open-source Python package NUBO (Diessner et al. 2023).

To make the comparison fair, the 30 runs used the same 30 initial starting points and random walks. For the random walks, this is achieved by sampling changes in percentages and scaling them by the maximal step size a rather than sampling absolute values directly.

Most physical experiments in engineering can only be conducted by introducing some noise, such as measurement uncertainty, that cannot be eliminated entirely. This section replicates this situation by adding randomly generated noise ϵ to the Hartmann function. The noisy function can be defined as $g(\mathbf{x}) = f(\mathbf{x}) + \epsilon$, where $f(\mathbf{x})$ is a deterministic negated Hartmann function from Section 4.2. The noise is sampled from a Normal distribution centered around zero with a small standard deviation σ , such that $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The simulations explore noise levels with $\sigma = 0.00$, $\sigma = 0.025$, $\sigma = 0.050$, and $\sigma = 0.100$. Considering the range of the Hartmann function, this corresponds to standard deviations of 0.75%, 1.5%, and 3.0% of the full output range, respectively. This means that for any of these three cases, 68.3% of the added noise values will fall between $\pm 1\sigma$, 95.5% fall between $\pm 2\sigma$ and 99.7% fall between $\pm 3\sigma$. For

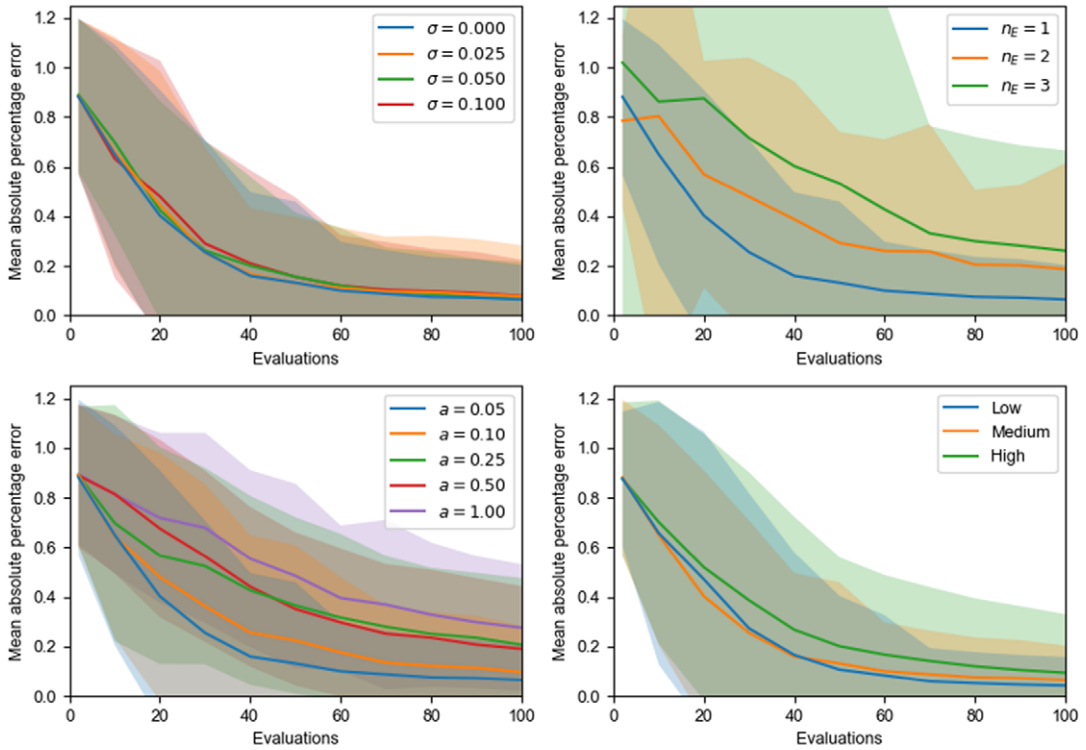


Figure 6. Means (lines) and 95% confidence intervals (shaded areas) of the mean absolute percentage error between the predictive mean of the Gaussian process and the truth over 30 replications for the six-dimensional Hartmann function. Upper-left: Comparison of randomly added noise levels, $\mathcal{N}(0, \sigma^2)$. Upper-right: Comparison of different numbers of uncontrollable parameters n_E . Lower-left: Comparison of five different step sizes a for the random walk $\mathcal{U}_{[-a,a]}$ added to the previous uncontrollable value. Lower-right: Comparison of uncontrollable variables with different parameter variability.

$\sigma = 0.100$, this translates into noise values that decrease or increase the real output by up to 3.0% of the output range 68.3% of the time, by 6.0% of the output range 95.5% of the time, and by 9.0% of the output range 99.7% of the time. The plot in the upper-left of Figure 6 shows the performance of ENVBO for each of these four noise levels. The results indicate no significant difference in the average performance or the 95% confidence intervals between the four cases. Overall, the proposed method is not sensitive to adding modest noise levels.

For some experiments, there might be more than one influential uncontrollable variable. The upper-right plot of Figure 6 provides results about the performance of ENVBO with one, two and three uncontrollable variables. At the same time, the overall dimensionality of the problem stays the same at $n = 6$. For $n_E = 1$, input six of the Hartmann function from Equation (4.2) is assumed uncontrollable, while input one is added to the uncontrollable variables for $n_E = 2$, and input one and four are added for $n_E = 3$. The number of test points used to evaluate the final Gaussian process model is increased from 25 to 50 for $n_E = 2$ and 75 for $n_E = 3$. The results show that the mean absolute percentage error increases with increasing numbers of environmental variables. Particularly, the 95% confidence intervals widen significantly. This result is expected as the input space of the environmental variables grows exponentially with n_E and requires exponentially more training points to cover the input space equally well as lower n_E . Thus, more evaluations are required to achieve similar results.

Uncontrollable variables will fluctuate to different extents from one evaluation to the next, for example, when measured in physical experiments. Higher fluctuations cause big jumps in the uncontrollable variable

values, while uncontrollable variables will be more stable for lower fluctuations. The lower-left plot of Figure 6 shows the results for five different fluctuation levels implemented by varying parameter a of the uniform distribution in Equation (14)—the higher a , the higher the fluctuation of the uncontrollable variable. The results show that Algorithm 2 performs better for lower a , and the performance of the final Gaussian process models decreases with increasing fluctuation.

Parameter variability indicates how much the output changes when moving along a parameter's input axis. Uncontrollable variables with a low variability will only change the output slightly, while variables with a high variability will change the output considerably. The effect of value changes of two variables could differ considerably if they have different levels of parameter variability. As a proxy for the parameter variability, the length scales of a well-fitting Gaussian process over the full domain are considered. The length scales quantify how long the outputs are correlated when moving along the input axes (Gramacy 2020). Consider, for example, a parameter with a large length scale. When this parameter value is changed by a certain amount, a relatively small change is expected in the output—provided everything else stays the same. The change in the output is expected to be more significant for a parameter with a small length scale. A Gaussian process is fitted to 2000 data points sampled from a Latin hypercube (McKay et al. 1979) to achieve a well-fitting model. The resulting length scales for all six parameters in order are 1.30, 1.90, 4.81, 1.48, 1.51, and 1.47. The first input has the lowest length scale, suggesting that outputs are only correlated for a short distance when moving along its axis. This means that the parameter variability of the first input is high. In contrast, the third input has the highest length scale, indicating a low parameter variability. Moving along its axis, less change is expected for the third input than for the first input. The lower-right plot of Figure 6 compares the first (high), third (low), and sixth input (medium) when chosen as the uncontrollable variable. Differences in the low and medium parameter variability are minimal. At the same time, there is some difference compared to the parameter with a high variability: the confidence interval is noticeably more significant, and the average of the mean absolute percentage error is slightly worse, especially for evaluations 30–80.

Finally, the relationship between the size of the effective domain, that is, the actual searched input space of the environmental variables, and the performance of the Gaussian process is investigated. Figure 7 uses the same data as Figure 6 but plots the effective parameter domain against the mean absolute percentage error, which provides a proxy for the performance of the Gaussian process. Each point reflects one individual replication of the 30 performed replications. The trend lines in each plot show a positive relationship between the effective domain and the mean absolute percentage error. Small effective domains generally correspond to small mean absolute percentage errors, while large effective domains generally correspond to large mean absolute percentage errors. Only the upper-right plot that plots the effective domain for different numbers of uncontrollable parameters against the mean absolute percentage error shows almost no relationship.

6. Application to a wind farm simulator

The power generation of wind farms is highly dependent on the wind speed, the wind direction and the placement of the individual wind turbines within a particular area or site (Grady et al. 2005; Qureshi and Warudkar 2023). Most of the time, interest lies in finding the optimal wind turbine positions that maximize the energy production conditional on either constant or variable wind speeds and directions (Mosetti et al. 1994; Chen et al. 2013; Parada et al. 2017). Optimization algorithms such as TOPFARM⁴ can be used when function evaluations are cheap, while methods discussed in Section 2 (Williams et al. 2000; Swersky et al. 2013; Toscano-Palmerin and Frazier 2018) present a cost-effective alternative when function evaluations are expensive. However, these methods cannot find multiple solutions for different environmental conditions within one optimization run, which is the objective of this section. Specifically, this application aims to find the optimal wind turbine positions conditional on randomly changing wind

⁴ See <https://topfarm.pages.windenergy.dtu.dk/TopFarm2/index.html> for further details on TOPFARM.

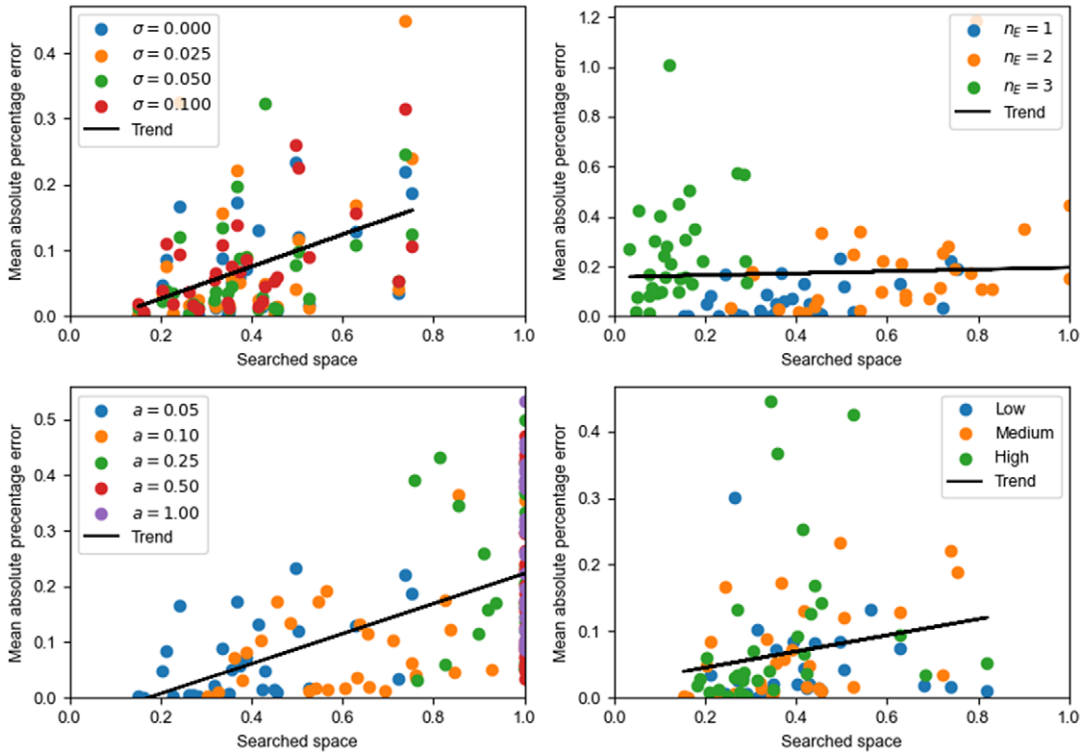


Figure 7. Relationships between the actual effective domain of the uncontrollable variables and the mean absolute percentage error of an individual run for the six-dimensional Hartmann function. Upper-left: Comparison of randomly added noise levels, $\mathcal{N}(0, \sigma^2)$. Upper-right: Comparison of different numbers of uncontrollable parameters n_E . Lower-left: Comparison of five different step sizes a for the random walk $U_{[-a, a]}$ added to the previous uncontrollable value. Lower-right: Comparison of uncontrollable variables with different parameter variability.

directions, resulting in one solution for each possible wind direction. While the wind speed is assumed to be fixed in this application, it could also be randomly changing. The result would be solutions for all combinations of wind direction and wind speed.

The first row of plots of Figure 8 shows the effect of the wind direction on the local wind speed for a complex underlying terrain while the global wind speed is fixed at 6 m/s. The locations of the high local wind speeds—necessary for high-energy production—shift significantly between a wind direction of 0 and 120 degrees. While there is a band of high local wind speeds down the center of the X location for the latter, it roughly rotates 90 degrees for the former. The plots show that ideal wind turbine positions will likely differ for the two wind directions. Another critical factor for maximizing energy production is taking the wake of the wind turbines into account (lower row of Figure 8). While the wake of the wind turbines located upstream does not affect wind turbines located downstream for a wind direction of 120, the wake of wind turbine 3 for a wind direction of 0 degrees heavily affects wind turbine 1. This shows that although a wind farm position can be ideal for one wind speed, it can be suboptimal for another.

In this section, a fictitious site is considered with complex terrain as shown in Figure 8 on which four wind turbines have to be placed to maximize the annual energy production (AEP). The wind direction is an environmental variable that varies according to a random walk as defined in Equation (14) where 90 and 135 degrees are the lower and upper bounds, respectively, and wind directions can change by ± 5 degrees from one iteration to the next. The global wind speed is fixed at 6 m/s. This results in a nine-dimensional problem with eight controllable parameters—one X and one Y location for each of the four

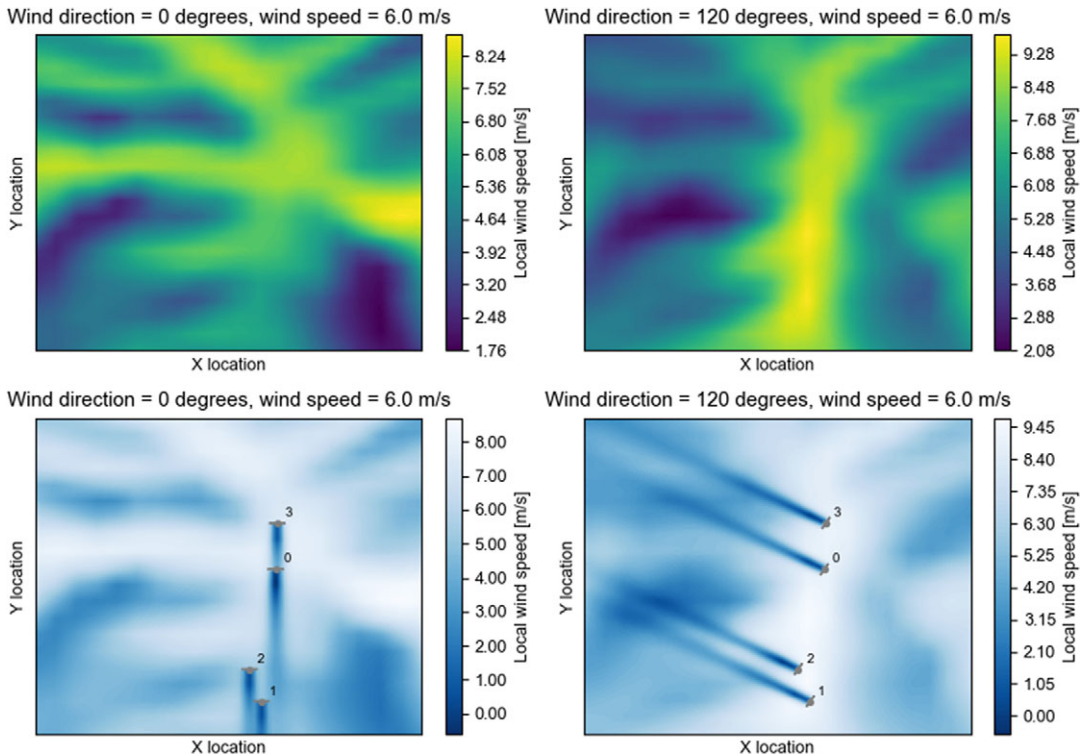


Figure 8. Wind farm simulator. Upper row: Local wind speed over the complex terrain for a wind direction of 0 and 120 degrees. Lower row: Wake of four wind turbines for a wind direction of 0 and 120 degrees. Wind speed is fixed at 6 m/s.

wind turbines—and the wind direction as the only environmental variable. Additionally, the positioning of the wind turbines is constrained such that wind turbines have to be at least 160 meters apart to prevent the blades from colliding. Simulations are performed with PyWake (Pedersen et al. 2023) and use Vestas V80 wind turbines that can produce 2 MW of energy and have a blade diameter of 80 meters. Simulating the power generation of wind farms is a complex task that requires an expensive simulator that accounts for many different parameters. However, the main objective of this example is to illustrate the performance of ENVBO and enable other researchers to replicate and validate this work. Thus, a relatively simple simulator was chosen to make the results easily reproducible so that ENVBO can be used with confidence on more expensive physical experiments and computer simulators.

ENVBO—a version of Algorithm 2 with EI as its acquisition function—is run for a function evaluation budget of 200 and benchmarked against regular Bayesian optimization (Algorithm 1) and the SLSQP optimization algorithm (Kraft 1994). Regular Bayesian optimization, referred to as BO in the following paragraphs, uses EI as its acquisition function to make comparisons fair. While ENVBO can return one solution for each wind direction after one optimization run, BO and the SLSQP algorithm must be run for each possible wind direction. To compare the algorithms, four wind directions are chosen—90, 105, 120, and 135 degrees—and BO is restricted to 50 function evaluations each to reach the same function evaluation budget as ENVBO. The function evaluations of the SLSQP algorithm cannot be restricted, and the algorithm is therefore run until convergence. BO and SLSQP are run for fixed wind directions, so they do not use the random walk in contrast to ENVBO. This makes converging toward a solution easier, as the algorithms can control all influential variables. ENVBO and BO were implemented via the open-source package NUBO (Diessner et al. 2023), and an off-the-shelf version of ENVBO is available at www.nubopy.com. The proposed SLSQP algorithm was implemented via the SciPy package

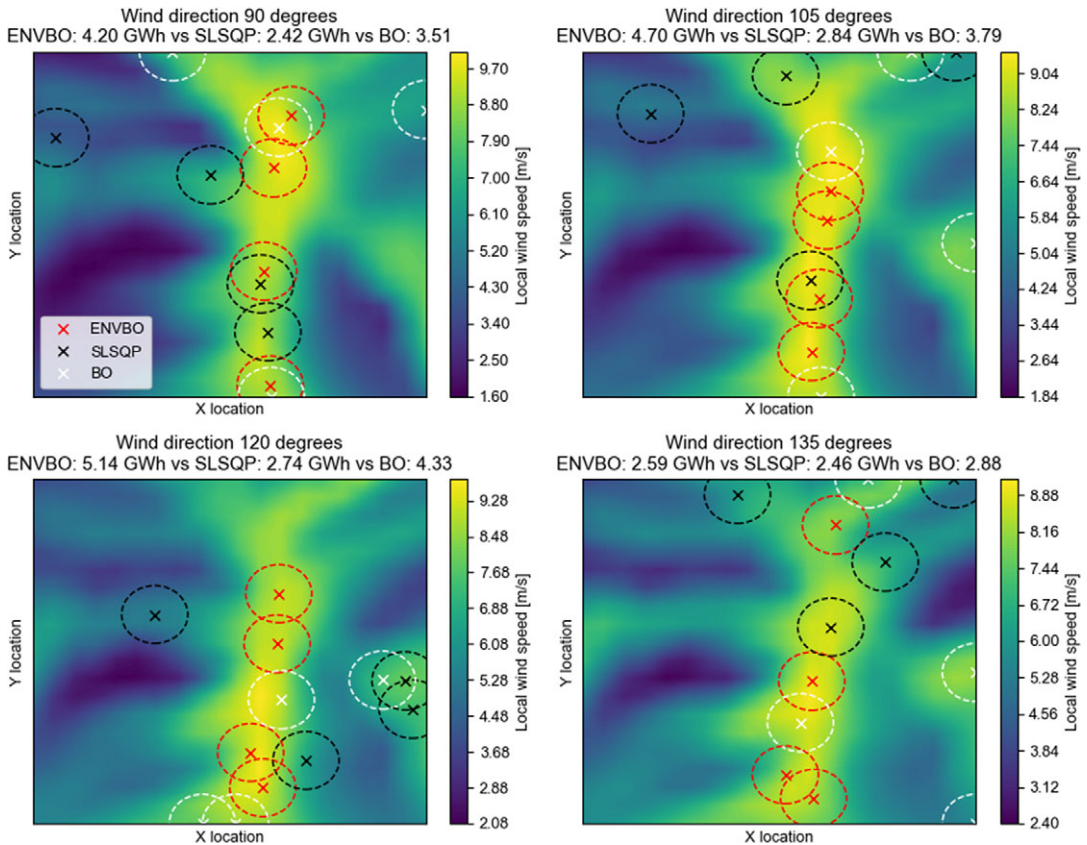


Figure 9. Annual energy production and placement of four wind turbines with spacing constraints. ENVBO (Algorithm 2) is benchmarked against SLSQP and BO (Algorithm 1).

(Virtanen et al. 2020). Furthermore, all code for optimizing the wind farm simulator is available at <https://github.com/mikediessner/environmental-conditions-BO>.

Figure 9 shows the results for all three algorithms and all four wind directions. The dashed circles around the wind turbine positions indicated with crosses represent the placement constraint—no wind turbine of one color can be placed within the dashed circle of another wind turbine with the same color. For a wind direction of 90 degrees, ENVBO performs best with an annual power production of 4.20 GWh, followed by BO and SLSQP with 3.51 and 2.42 GWh, respectively. This is a 20% improvement over BO and a 74% improvement over SLSQP. SLSQP performs much worse than ENVBO and places at least one turbine in a subpar area with low local wind speeds, possibly due to converging toward a local maximum. While BO places three turbines in areas with high local wind speeds, it places one within a suboptimal area. For a wind direction of 105 degrees, ENVBO with 4.70 GWh performs significantly better than BO and SLSQP, achieving 0.91 and 1.86 GWh less power generation, respectively—a 24% and 65% improvement. A similar result is achieved for a wind direction of 120 degrees. ENVBO outperforms BO and SLSQP by 0.81 and 2.4 GWh or 19% and 88%, respectively. The results of the three strategies are closest for a wind direction of 135 degrees. This is the only instance where a benchmark beats ENVBO—in this case, BO with an AEP of 2.88 GWh. ENVBO achieves 0.29 GWh (10%) less AEP but still outperforms SLSQP by 0.13 GWh (5%). However, the differences are much smaller than for any of the first three wind directions. Considering Figure 9, ENVBO is the only strategy that consistently places all four wind turbines in the band of high local wind speeds located down the center of the X location. At least one turbine is placed off to one side of this band for all other methods.

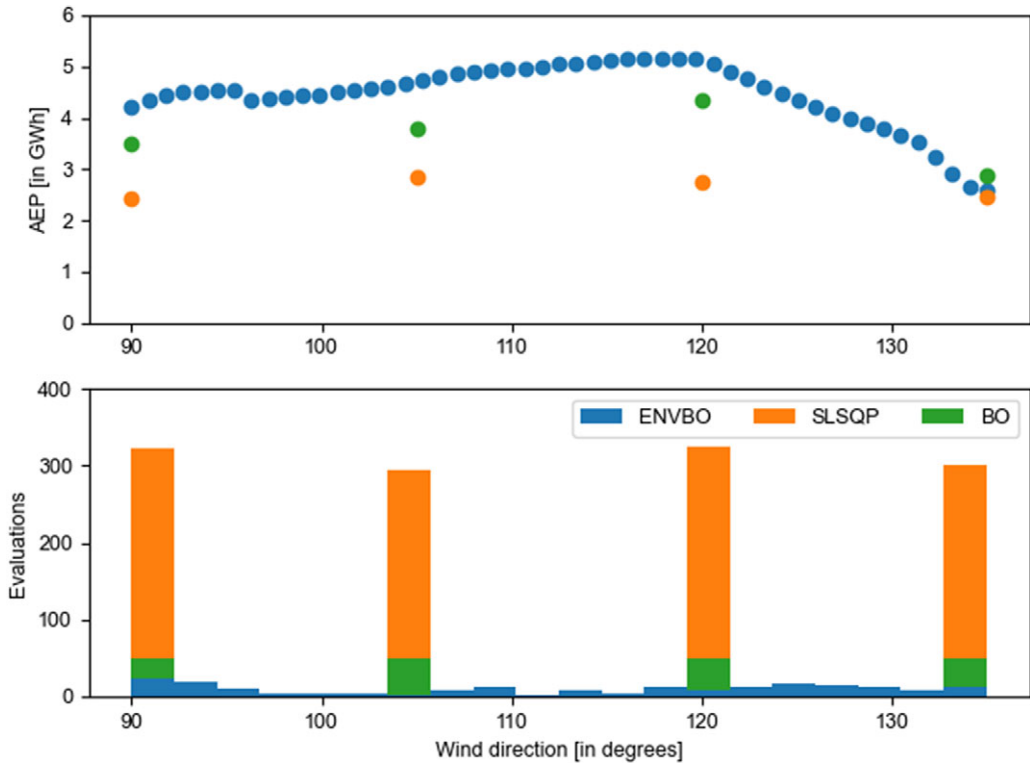


Figure 10. Annual energy production and number of function evaluations conditional on the wind direction. ENVBO (Algorithm 2) is benchmarked against SLSQP and BO (Algorithm 1).

While the results in Figure 9 show that ENVBO outperforms BO and SLSQP in almost all instances, they do not consider the different numbers of function evaluations required by each algorithm. Figure 10 plots the AEP and the number of function evaluations against the wind direction. The lower plot shows that SLSQP uses the most function evaluations by far, with 323, 294, 324, and 301 evaluations for the four wind directions—a total budget of 1152 function evaluations. In contrast, ENVBO and BO both use 200 function evaluations in total. BO divides this budget equally over the four wind directions, allocating 50 evaluations per wind direction, while they are distributed via a random walk for ENVBO. Each of ENVBO's bins of the lower plot only contains two to 24 function evaluations. Compared to SLSQP, both Bayesian optimization algorithms are much more sample-efficient and use less than 20% of its evaluation budget. ENVBO uses less than half the evaluation for the four wind directions compared to BO, but it outperforms BO for three of the four wind directions, as shown in the upper plot. This shows the advantage of a global surrogate model that is fit to all controllable and environmental variables. The Gaussian process uses all available information and can model the effect of the environmental conditions.

Although the improved performance of ENVBO compared to BO and SLSQP is already valuable, the main advantage lies in ENVBO's capability to predict a solution for each possible wind direction. This is illustrated by the 51 solutions given for ENVBO in the upper plot of Figure 10. SLSQP and BO can only give solutions for the specific wind direction for which they were run. To achieve similar results, SLSQP and BO need to be rerun for each wind direction, which would multiply the required function evaluations many times. ENVBO uses the available function evaluation budget much more effectively and is a more sample-efficient and cost-effective approach.

Overall, ENVBO finds solutions over the whole range of the environmental variable, while in most cases performing up to 88% better than algorithms that focus on one fixed environmental variable at a

time. This is even true for ranges of the environmental variable that are only explored briefly as the algorithm learns the effect of the environmental variable from adjacent areas. Furthermore, ENVBO uses only a small fraction of the evaluation budget of the two benchmarks, making it particularly beneficial for optimizing computer simulators and physical experiments that can be very expensive to run in engineering.

7. Discussion

This section discusses the empirical results of Section 5 and the results of the application to a simple wind farm simulator in Section 6 by interpreting the results, considering limitations and implications, and highlighting the use for the scientific community.

The effective optimization of noisy objective functions in engineering applications has been considered from the earliest studies in Bayesian optimization by Kushner (1964) and later by Jones et al. (1998) through their EGO algorithm, emphasizing the importance of optimizing noisy objective functions in engineering. Indeed, most experiments cannot be conducted without noise, and many simulators are stochastic. Gramacy and Lee (2010) argue that even deterministic simulators can be seen as noisy as they are an approximation of the real world. Noise can be understood as the discrepancy between computer code and reality. This makes investigating the robustness of a proposed approach to noise essential. The empirical investigation in this article showed that no difference is noticeable between the average performance of ENVBO for deterministic and stochastic objective functions. The results show that ENVBO can optimize noisy objective functions, making it applicable to many experiments and simulators prevalent in engineering. However, the noise added to the six-dimensional Hartmann function was lower than 9% of the output variable range and assumed to be homoscedastic. Thus, one should be cautious when applying ENVBO to experiments with very high or heteroscedasticity noise. A solution for the latter case could be to use a heteroscedastic Gaussian process for the surrogate modeling, which uses a second Gaussian process to model the noise (Goldberg et al. 1997).

The results regarding the number of uncontrollable parameters in Section 5 show that the performance decreases with an increasing number of uncontrollable parameters. This decrease in performance is reflected in higher mean absolute percentage errors and higher variability between replications. For example, the mean absolute percentage error after 100 evaluations is 12.2% and 19.6% higher for $n_d = 2$ and $n_d = 3$, respectively, than for $n_d = 1$. These results are not surprising since the uncontrollable input space grows exponentially with the number of environmental variables, making it more challenging to achieve even coverage over the uncontrollable space due to the randomness of the environmental variables. Furthermore, the problem of extrapolation with Gaussian processes increases with the growing number of dimensions of the uncontrollable input space. The test points used to evaluate the final prediction model given by ENVBO are generated with a Latin hypercube that uses the minimal and maximal values of all environmental variables as their upper and lower bounds. With increasing numbers of the environmental variables n_E , it is very likely that while values for individual dimensions fall within these bounds, the combination of values for different dimensions falls in areas not explored by the optimization algorithm. The final Gaussian process model will extrapolate to predict outputs for these test points. As Section 4 discusses, extrapolation with Gaussian processes generally means that predictions default to the prior mean function. Gaussian processes with a more complex prior mean function can be explored (Planas et al. 2020) to counteract this issue. Moreover, the empirical investigation in this article assumes a single fluctuation level a for each environmental input. However, in a real-world application, different combinations of fluctuation levels are more realistic, which might require more function evaluations. In any case, the resulting prediction model must be carefully scrutinized to ensure a good fit and minimal extrapolation.

This work assumes that the environmental variables cannot be controlled and are randomly selected according to the uniform random walk in Equation (14). Thus, it is vital to scrutinize the performance of ENVBO based on this assumption and to consider different levels of fluctuation and parameter variability. Environmental variables are likely to fluctuate to different degrees. For example,

temperature is likely to change more slowly than wind speed. Section 5 studies this by considering different levels of fluctuation and parameter variability, which are both closely connected and might be challenging to distinguish in practice. Fluctuation, in our case, is defined as the size of a , the constant of the uniform random walk. Parameter variability is defined as the effect on the output for a given change of the environmental variable. For higher fluctuations, there is more potential for larger effective domains of the uncontrollable variable, that is, the range for which the environmental variables are explored. This can be explained by considering the parameter a . For $a = 1.00$, any domain value can be randomly selected at each optimization step, and the whole domain will likely be searched. For $a = 0.05$, only values that differ by 0.05 from the previous evaluation can be randomly selected, making it less likely that the whole domain will be searched before the evaluation budget is exhausted. Suppose a Gaussian process is fitted to domains with different sizes, but the number of evaluations remains the same. In that case, predictions will typically be better for smaller domains. For example, it is plausible that a Gaussian process fitted to 10 data points within the domain $[0, 0.1]$ will reflect the truth in this domain better than a Gaussian process fitted to 10 data points within the much larger domain $[0, 1]$ assuming that everything else stays comparable. Similarly, a parameter with a small correlation length when moving along its axis and for which small changes affect the output significantly will be much more challenging to model than a parameter correlated for longer distances. This is intuitive as a longer correlation length makes predicting the effect of changing the parameter easier. The results in Figure 6 show that the average performance and the variability in performance worsen with increasing fluctuation and parameter variability levels. Thus, the prediction reliability depends on these levels; higher levels require more function evaluations to perform similarly for environmental variables with less fluctuation. While these results seem intuitive, some limitations and caveats should be considered. First, the assumption that environmental variables change according to a uniform distribution will not generally be true. The uniform assumption can thus only be viewed as a proxy for the underlying distributions associated with arbitrary environmental variables. Second, as the environmental variables are selected randomly, the distribution of observed points could be skewed such that some regions will contain more observations than others. The final model will potentially be less accurate in regions with fewer observations. Hence, it makes sense to consider not only the prediction but also the associated uncertainty. If the uncertainty is considerable, we might not trust the prediction to the same extent. Finally, the randomness in the environmental variables means there could be cases where the environmental variable is almost constant in the observed sample, and we cannot explore most of its range. One must be cautious not to extrapolate in these instances.

The results of Figure 7 suggest an inverse relationship between the performance of ENVBO and the size of the effective domain—that is, the explored space of the environmental variables—where ENVBO performs better for smaller effective domains. Intuitively, this result makes sense as a Gaussian process should have a better fit for a smaller space than a larger space when the number of data points and the function in question stay the same. The results depicted in the lower-left plot reinforce the reasoning that the larger the fluctuation parameter of the uniform distribution a , the more potential there is for a larger parameter space to be explored. Thus, it is plausible that ENVBO performs better with smaller values of a as the results of the lower-left plot of Figure 6 suggest. Generally, the size of the environmental variable space grows exponentially with the number of environmental variables. Exponentially more evaluations are required to achieve the same coverage for $n_E = 3$ as for $n_E = 1$. However, in this study, the evaluation budget is fixed to 100 evaluations regardless of the number of environmental variables. Considering this, an inverse effect of the number of environmental variables and the effective domain would be expected. However, searched spaces for $n_E = 1$ are generally higher than for $n_E = 3$ but smaller than for $n_E = 2$. A possible reason for this is that the first environmental variable uses a fluctuation parameter $a = 0.05$ while the second and third environmental variables use $a = 0.1$. A change from $a = 0.05$ to $a = 0.1$ can significantly affect the domain size, as shown in the lower-left plot of Figure 7, and might be responsible for the unusual order of effective domain sizes. In this investigation, results suggest that the size of the effective domain can inform the number of function evaluations required, and larger effective domains indicate that more function evaluations are necessary.

While these results are from the six-dimensional Hartmann function and might not be generalizable to all objective functions, the application of ENVBO to a simple wind farm simulator presented in [Section 6](#) highlights the approach in an illustrative example. ENVBO outperforms the SLSQP algorithm in all cases and the standard Bayesian optimization algorithm in three out of four cases. ENVBO used fewer function evaluations per wind speed than SLSQP and standard Bayesian optimization and can make predictions for the full range of wind speeds compared to the benchmarks that can only make predictions for fixed wind speeds. These results suggest that ENVBO successfully leverages the correlation within the environmental variable, resulting in a superior and more sample-efficient algorithm. The reason standard Bayesian optimization performs better than ENVBO for a single wind speed is likely that it is at the upper limit of the environmental variable's range, and ENVBO can only leverage correlation for smaller wind speeds. [Figure 10](#) shows that very few function evaluations were observed for this upper limit with ENVBO, which was insufficient to yield a good model fit. As discussed previously, a more complex prior mean function could improve this result.

Although this wind farm simulator is a simple representation of a complex real-world problem, it has characteristics of many engineering problems. Together with the potential to optimize noisy objective functions, multiple environmental variables and different levels of parameter fluctuation and variability, ENVBO is a valuable tool for the broader scientific community. It also does not require information about the correlation structure that is leveraged by other methods (Krause and Ong 2011) and is available as an off-the-shelf algorithm as part of the Python package NUBO (Diessner et al. 2023) that can conveniently be run on cloud services such as Google Colaboratory without requiring a Python installation on a local machine.

8. Conclusion

When optimizing physical experiments, often only some variables can be fully controlled, or interest lies in finding not one global optimum but one optimum for each value of a particular variable—essentially a function that maps environmental variable values to optimal values for the controllable parameters. In the past, Bayesian optimization was used predominantly to find global optima. This study extends the Bayesian optimization algorithm to situations with changing environmental conditions and makes the following contributions:

1. Development of ENVBO, an algorithm for optimizing expensive black-box functions with randomly changing environmental conditions that does not require any knowledge about the objective function, particularly the environmental variables.
2. Analysis of ENVBO and its sensitivity to noisy objective functions, the number of environmental variables and different levels of fluctuation and parameter variability.
3. Implementation of ENVBO as part of the Python package NUBO.
4. Illustration of ENVBO on a simple wind farm simulator and comparison to standard Bayesian optimization and the SLSQP algorithm.

Compared to related research in this area and particularly the closest related work of Krause and Ong (2011), this article investigates and gives firm proposals and justification for the implementation of the algorithm and offers an easy-to-use implementation in Python. Furthermore, while Krause and Ong (2011) show how known (linear or additive) structures can be exploited via composite kernels, ENVBO can also be used where no information on the objective function and the environmental variables is available. Finally, ENVBO uses EI as its acquisition function to guide the selection of candidate points. Compared to UCB used in Krause and Ong (2011), EI has the advantage of not requiring the specification of a trade-off parameter that balances exploitation and exploration and has the potential to influence the effectiveness of the algorithm.

The proposed method fits a global surrogate model over all controllable and environmental variables and uses measurements of the environmental variables to conditionally optimize the acquisition function

with regard to the controllable parameters. This conditional optimization enables finding a model with a posterior predictive mean that provides close to optimal values of the controllable parameters for any values of the uncontrollable variables. The original Bayesian optimization algorithm assumes that the uncontrollable variables are fixed or disregarded entirely. Achieving similar results requires repeating the optimization process for various fixed values for the uncontrollable variables and interpolating between the found optima. Thus, the proposed approach is more sample-efficient, using all available information about the objective function in one optimization run.

For the proposed optimization strategy presented in [Algorithm 2](#), three different acquisition functions—EI, LogEI, and UCB with different trade-off parameters β —were considered, and their performance compared on two synthetic test functions—the two-dimensional Levy function and the six-dimensional Hartmann function. EI performed best across the two test functions and was thus chosen as the acquisition function for the proposed ENVBO algorithm for the case study. Furthermore, this study empirically investigated the properties of ENVBO, showing that the algorithm is effective in the scenarios of added noise and uncontrollable variables with high and low variability. When solving problems with more than one environmental variable, it has to be ensured that the final Gaussian process model is not used for extrapolation, as this will result in biased solutions. Additionally, it was found that uncontrollable variables with large fluctuations require more function evaluations than uncontrollable variables that fluctuate less. Higher fluctuation generally results in a larger effective parameter domain than lower fluctuation. When modeling, it is intuitive that larger areas require more observations—and thus information—than smaller areas to achieve identical results, assuming that all other properties are comparable.

ENVBO—an implementation of the proposed algorithm within the Python package NUBO (Diessner et al. 2023)—was applied to a wind farm simulator with the objective to place four wind turbines within an area with complex underlying terrain to find positions that maximize the annual power generation for different wind directions. The results were compared to two benchmarks—regular Bayesian optimization via NUBO and the SLSQP algorithm via the SciPy package (Virtanen et al. 2020)—showing up to 88% better performance in all, but one case across the whole range of possible wind directions while keeping function evaluations and thus costs low. ENVBO is a sample-efficient and cost-effective approach for optimizing expensive experiments and simulators with uncontrollable environmental conditions.

In the future, we plan to apply ENVBO to more complex simulators and experiments and investigate how more complex mean functions can be used in the prior distribution of the Gaussian process to better represent areas of the input space where only a small number of data points are available—and also to give a better representation for areas toward the edges of the ranges of the environmental variables. [Section 7](#) discussed another possible extension of ENVBO to allow optimization with heteroscedastic noise through implementing heteroscedastic Gaussian processes (Goldberg et al. 1997). Furthermore, an interesting research direction is finding a method to evaluate if ENVBO should observe a point from the objective function for a given environmental variable value. There could be situations where enough data points have already been collected for an environmental condition, and we should refrain from observing another data point as it would only add very limited additional information. It might be more sample-efficient to wait for an environmental condition that has not been explored. Finally, we implicitly assume that the best observation of all previous training data points is a good target for the EI function in [Equation \(10\)](#). However, improving upon the best-observed output for any given environmental condition might not be possible. Consider the case where the single global optimum has been found, but we are still looking for local optima for different environmental conditions. No improvement would be possible in this case, and the acquisition function would suggest nonoptimal candidates. A possible solution is to use the maximum posterior prediction for the given environmental condition as the target. This is similar to what Picheny et al. (2013) and Gramacy (2020) propose in the presence of noise.

Data availability statement. Replication data and code can be found in the GitHub repository at <https://github.com/mikediessner/environmental-conditions-BO> and <https://doi.org/10.5281/zenodo.10619544>.

Author contribution. Conceptualization: M.D., K.J.W., and R.D.W. Data curation: M.D. Formal analysis: M.D. Funding acquisition: K.J.W. and R.D.W. Investigation: M.D. Methodology: M.D. Software: M.D. Supervision: K.J.W. and R.D.W. Validation: M.D. Visualization: M.D. Writing—original draft: M.D. Writing—review and editing: M.D., K.J.W., and R.D.W.

Funding statement. The work has been supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/T020946/1 and the EPSRC Center for Doctoral Training in Cloud Computing for Big Data under grant number EP/L015358/1.

Competing interest. None declared.

Ethical standard. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

References

- Ament S, Daulton S, Eriksson D, Balandat M and Bakshy E (2023) Unexpected improvements to expected improvement for Bayesian optimization. *Advances in Neural Information Processing Systems* 37.
- Auer P (2003) Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, 397–422.
- Belakaria S Deshwal, A and Doppa JR (2020). Max-value entropy search for multi-objective bayesian optimization. *Advances in Neural Information Processing Systems* 32.
- Bergstra J, Bardenet R, Bengio Y and Kégl B (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems* 24.
- Berry DA and Fristedt B (1985) *Bandit Problems: Sequential Allocation of Experiments* (Monographs on Statistics and applied probability). London: Chapman & Hall.
- Brochu E, Cora VM and de Freitas N (2010) A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *ArXiv*, abs/1012.2599.
- Bull AD (2011) Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research* 12, 2879–2904.
- Calandra R, Seyfarth A, Peters J and Deisenroth MP (2015) Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence* 76, 5–23.
- Chang PB, Williams BJ, Bhalla KSB, Belknap TW, Santner TJ, Notz WI and Bartel DL (2001) Design and analysis of robust total joint replacements: finite element model experiments with environmental variables. *Journal of Biomechanical Engineering* 123(3), 239–246.
- Chang PB, Williams BJ, Santner TJ, Notz WI and Bartel DL (1999) Robust optimization of total joint replacements incorporating environmental variables. *Journal of Biomechanical Engineering* 121(3), 304–310.
- Char I, Chung Y Neiswanger W, Kandasamy K, Nelson AO, Boyer M, Kolemen E and Schneider J (2019) Offline contextual Bayesian optimization. *Advances in Neural Information Processing Systems* 32.
- Chen Y, Li H, Jin K and Song Q (2013) Wind farm layout optimization using genetic algorithm with different hub height wind turbines. *Energy Conversion and Management* 70, 56–65.
- Chung Y, Char I, Neiswanger W, Kandasamy K, Nelson AO, Boyer, M, Kolemen, E and Schneider J (2020) Offline contextual Bayesian optimization for nuclear fusion. *ArXiv*, abs/2001.01793.
- Diessner M, O'Connor J, Wynn A, Laizet S, Guan Y, Wilson K and Whalley RD (2022) Investigating Bayesian optimization for expensive-to-evaluate black box functions: application in fluid dynamics. *Frontiers in Applied Mathematics and Statistics* 8, 1076296.
- Diessner M, Wilson K and Whalley RD (2023) NUBO: a transparent Python package for Bayesian Optimisation. *arXiv preprint arXiv:2305.06709*. Accepted at *Journal of Statistical Software*.
- Duris J, Kennedy D, Hanuka A, Shtalenkova J, Edelen A, Egger A, Cope T and Ratner D (2019) Bayesian optimization of a free-electron laser. *Physical Review Letters* 124(12), 124801.
- Emmerich M, Giannakoglou KC and Naujoks B (2006) Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodells. *IEEE Transactions on Evolutionary Computation* 10, 421–439.
- Fernández-Sánchez D, Garrido-Merchán E and Hernández-Lobato D (2023) Improved max-value entropy search for multi-objective Bayesian optimization with constraints. *Neurocomputing* 546, 126290.
- Frazier P (2018) A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*.
- Frazier P and Powell WB (2007) The knowledge gradient policy for offline learning with independent normal rewards. *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning* 2007, 143–150.
- Frazier P, Powell WB and Dayanik S (2009) The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing* 21, 599–613.
- Frazier P and Wang J (2015) Bayesian optimization for materials design. *arXiv: Machine Learning*.
- Garnett R (2023) *Bayesian optimization*. Cambridge: Cambridge University Press.
- Garnett R, Osborne MA and Roberts SJ (2010) Bayesian optimization for sensor set selection. In *International Symposium on Information Processing in Sensor Networks* 9, 209–219.

- Ginsbourger D, Baccou J, Chevalier C, Perales F, Garland N and Monerie Y** (2014) Bayesian adaptive reconstruction of profile optima and optimizers. *SIAM/ASA Journal on Uncertainty Quantification* 2(1), 490–510.
- Goldberg PW, Williams CKI and Bishop CM** (1997) Regression with input-dependent noise: a Gaussian process treatment. In *Advances in Neural Information Processing Systems*.
- Grady S, Hussaini MY and Abdullah MM** (2005) Placement of wind turbines using genetic algorithms. *Renewable Energy* 30, 259–270.
- Gramacy RB** (2020) *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*, 1st Edn., Boca Raton, FL:Chapman & Hall/CRC. <http://bobby.gramacy.com/surrogates/>.
- Gramacy RB, Gray GA, Digabel SL, Lee HKH, Ranjan P, Wells G and Wild SM** (2016) Modeling an augmented lagrangian for blackbox constrained optimization. *Technometrics* 58(1), 1–11.
- Gramacy RB and Lee HKH** (2010) Cases for the nugget in modeling computer experiments. *Statistics and Computing* 22, 713–722.
- Groot P, Birlutiu A and T H** (2010) Bayesian Monte Carlo for the global optimization of expensive functions. *Frontiers in Artificial Intelligence and Applications* 215, 249–254.
- Hennig P and Schuler CJ** (2011) Entropy search for information-efficient global optimization. *ArXiv*, abs/1112.1217.
- Hernández-Lobato JM, Gelbart MA, Hoffman MW, Adams RP and Ghahramani Z** (2015) Predictive entropy search for bayesian optimization with unknown constraints. In *International Conference on Machine Learning*.
- Hernández-Lobato JM, Hoffman MW and Ghahramani Z** (2014) Predictive entropy search for efficient global optimization of black-box functions. *ArXiv*, abs/1406.2541.
- Hutter F, Hoos HH and Leyton-Brown K** (2011) Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*.
- Jones DR** (2001) A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization* 21, 345–383.
- Jones DR, Schonlau M and Welch WJ** (1998) Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13(4), 455.
- Kraft D** (1994) Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS)* 20(3), 262–281.
- Krause A and Ong C** (2011) Contextual Gaussian process bandit optimization. *Advances in Neural Information Processing Systems* 24.
- Kushner HJ** (1962) A versatile stochastic model of a function of unknown and time varying form. *Journal of Mathematical Analysis and Applications* 5(1), 150–167.
- Kushner HJ** (1964) A new method of locating the maximum point of an arbitrary multiplex curve in the presence of noise. *Journal of Basic Engineering* 86(1), 97–106.
- Lattimore T and Szepesvári C** (2020) *Bandit algorithms*. Cambridge: Cambridge University Press.
- Li Y Reyes KG, Vazquez-Anderson J, Wang Y, Contreras LM and Powell WB** (2018) A knowledge gradient policy for sequencing experiments to identify the structure of ma molecules using a sparse additive belief model. *INFORMS Journal on Computing* 30(4), 750–767.
- Lindauer M, Eggenberger K, Feurer M, Biedenkapp A, Deng D, Benjamins C, Ruhkopf T, Sass R and Hutter F** (2022) Smac3: a versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research* 23 (54), 1–9.
- Lyu W, Xue P, Yang F, Yan C, Hong Z, Zeng X and Zhou D** (2018) An efficient Bayesian optimization approach for automated optimization of analog circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers* 65, 1954–1967.
- Mahfoze OA, Moody A, Wynn A, Whalley RD and Laizet S** (2019) Reducing the skin-friction drag of a turbulent boundary-layer flow with low-amplitude wall-normal blowing within a bayesian optimization framework. *Physical Review Fluids* 4(9), 094601.
- Mallor F, Semprini-Cesari G, Mukha T, Rezaeiravesh S and Schlatter P** (2024) Bayesian optimization of wall-normal blowing and suction-based flow control of a NACA 4412 wing profile. *Flow, Turbulence and Combustion* 113(1), 93–118.
- Maraval AM, Zimmer M, Grosnit A, Tutunov R, Wang J and Ammar H** (2022) Sample-efficient optimisation with probabilistic transformer surrogates. *ArXiv*, abs/2205.13902.
- Martinez-Cantin R, de Freitas N, Brochu E, Castellanos JA and Doucet A** (2009) A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots* 27, 93–103.
- McKay MD, Beckman RJ and Conover WJ** (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21(2), 239–245.
- Močkus J** (1972) Bayesian methods of search for an extremum. *Automatika i Vychislitel'naya Tekhnika (Automatic Control and Computer Sciences)* 6(3), 53–62.
- Močkus J** (1974) On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference: Novosibirsk*, July 1–7, 1974, volume 27, pp. 400–404. Springer-Verlag.
- Močkus J** (1989) *Bayesian Approach to Global Optimization: Theory and Applications*, vol. 37 of *Mathematics and its Applications*. Dordrecht: Springer-Verlag.
- Mosetti G, Poloni C and Diviacco B** (1994) Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm. *Journal of Wind Engineering and Industrial Aerodynamics* 51, 105–116.

- Neal RM (1996) *Bayesian Learning for Neural Networks*, vol. 118 of *Lecture Notes in Statistics*, 1st Edn., New York: Springer-Verlag
- Negoescu DM, Frazier P and Powell WB (2011). The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing* 23, 346–363.
- O'Connor J, Diessner M, Wilson K, Whalley RD, Wynn A and Laizet S (2023) Optimisation and analysis of streamwise-varying wall-normal blowing in a turbulent boundary layer. *Flow, Turbulence and Combustion* 110, 993–1021.
- Packwood D (2017) *Bayesian Optimization for Materials Science*. Singapore: Springer Singapore.
- Parada L, Herrera C, Flores P and Parada V (2017) Wind farm layout optimization using a Gaussian-based wake model. *Renewable Energy* 107, 531–541.
- Pearce M and Branke J (2017) Efficient Information Collection on Portfolios. Working Paper. University of Warwick.
- Pearce M and Branke J (2018) Continuous multi-task Bayesian optimisation with correlation. *European Journal of Operational Research* 270(3), 1074–1085.
- Pedersen MM, Forsting AM, van der Laan P, Riva R, Romàn LAA, Risco JC, Friis-Møller M, Quick J, Christiansen JPS, Rodrigues RV, Olsen BT and Réthoré P-E (2023) Pywake 2.5.0: An Open-Source Wind Farm Simulation Tool. DTU Wind, Technical University of Denmark.
- Picheny V, Ginsbourger D, Richet Y and Caplin G (2013) Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 55, 13–12.
- Planas R, Oune N and Bostanabad R (2020) Extrapolation with Gaussian Random Processes and Evolutionary Programming. *Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 11A: 46th Design Automation Conference (DAC)*, 37.
- Ponweiser W, Wagner T, Biermann D and Vincze M (2008) Multiobjective optimization on a limited budget of evaluations using model-assisted-metric selection. *Parallel Problem Solving from Nature* 10, 784–794.
- Qureshi TA and Warudkar V (2023) Wind farm layout optimization through optimal wind turbine placement using a hybrid particle swarm optimization and genetic algorithm. *Environmental Science and Pollution Research* 30, 77436–77452.
- Rasmussen CE and Williams CKI (2006) *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press.
- Šaltenis VR (1971) One method of multiextremum optimization. *Avtomatika i Vychislitel'naya Tekhnika (Automatic Control and Computer Sciences)* 5(3), 33–38.
- Schonlau M (1997) Computer Experiments and Global Optimization. PhD thesis, University of Waterloo.
- Scott WR, Frazier P and Powell WB (2011) The correlated knowledge gradient for simulation optimization of continuous parameters using Gaussian process regression. *SIAM Journal on Optimization* 21, 996–1026.
- Shahriari B, Swersky K, Wang Z, Adams RP and De Freitas N (2015) Taking the human out of the loop: a review of Bayesian optimization. *Proceedings of the IEEE* 104(1), 148–175.
- Snoek J, Larochelle H and Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems* 25.
- Snoek J, Rippel O, Swersky K, Kiros R, Satish N, Sundaram N, Patwary MMA, and Prabhat Adams RP (2015) Scalable Bayesian optimization using deep neural networks. *International Conference on Machine Learning*.
- Springenberg JT, Klein A, Falkner S and Hutter F (2016) Bayesian optimization with robust Bayesian neural networks. *Advances in Neural Information Processing Systems* 29.
- Srinivas N, Krause A, Kakade S and Seegre M (2010) Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the International Conference on Machine Learning, 2010*.
- Sterling D, Sterling T, Zhang Y and Chen H (2015) Welding parameter optimization based on gaussian process regression bayesian optimization algorithm. *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1490–1496.
- Swersky K, Snoek J and Adams RP (2013) Multi-task Bayesian optimization. *Advances in Neural Information Processing Systems* 26.
- Tiao LC, Klein A, Seeger MW, Bonilla EV, Archambeau C and Ramos FT (2021) Bore: Bayesian optimization by density-ratio estimation. In *International Conference on Machine Learning*.
- Toscano-Palmerin S and Frazier P (2018) Bayesian Optimization with Expensive Integrands. *arXiv preprint arXiv:1803.08661*.
- Villemonteix J, Vázquez E and Walter E (2006) An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization* 44, 509–534.
- Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P and SciPy 1.0 Contributors (2020) SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* 17, 261–272.
- Wang Z and Jegelka S (2017) Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pp. 3627–3635. PMLR.
- Williams BJ, Santner TJ and Notz WI (2000) Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica* 10(4), 1133–1152.

- Xie J, Frazier P, Sankaran S, Marsden AL and Elmohamed S** (2012) Optimization of computationally expensive simulations with gaussian processes and parameter uncertainty: application to cardiovascular surgery. *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 406–413.
- Yang K, Emmerich M, Deutz AH and Bäck T** (2019a) Efficient computation of expected hypervolume improvement using box decomposition algorithms. *Journal of Global Optimization* 75, 3–34.
- Yang K, Emmerich M, Deutz AH, and Bäck, T** (2019b) Multi-objective Bayesian global optimization using expected hypervolume improvement gradient. *Swarm and Evolutionary Computation* 44, 945–956.
- Zhu C, Byrd RH, Lu P and Nocedal J** (1997) Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)* 23(4), 550–560.
- Žilinskas A** (1975) Single-step Bayesian search method for an extremum of functions of a single variable. *Cybernetics* 11(1), 160–166.

Cite this article: Diessner M, Wilson KJ and Whalley RD (2024). On the development of a practical Bayesian optimization algorithm for expensive experiments and simulations with changing environmental conditions. *Data-Centric Engineering*, 5, e45. doi:10.1017/dce.2024.40