Review Article

Bayesian optimization algorithms for accelerator physics

Ryan Roussel,[1,*] Auralee L. Edelen,[1] Tobias Boltz,[1] Dylan Kennedy,[1] Zhe Zhang,[1] Fuhao Ji,[1] Xiaobiao Huang,[1] Daniel Ratner,[1] Andrea Santamaria Garcia,[2] Chenran Xu,[2] Jan Kaiser,[3] Angel Ferran Pousa,[3] Annika Eichler,[3,4] Jannis O. Lübsen,[4] Natalie M. Isenberg,[5] Yuan Gao,[5] Nikita Kuklev,[6] Jose Martinez,[6] Brahim Mustapha,[6] Verena Kain,[7] Christopher Mayes,[8] Weijian Lin,[9] Simone Maria Liuzzo,[10] Jason St. John,[11] Matthew J. V. Streeter,[12] Remi Lehe,[13] and Willie Neiswanger[14]

[1]*SLAC National Laboratory, Menlo Park, California 94025, USA*
[2]*Karlsruhe Institute of Technology, Karlsruhe, Germany*
[3]*Deutsches Elektronen-Synchrotron DESY, Germany*
[4]*Hamburg University of Technology, 21073 Hamburg, Germany*
[5]*Brookhaven National Laboratory, Upton, New York 11973, USA*
[6]*Argonne National Laboratory, Lemont, Illinois 60439, USA*
[7]*European Organization for Nuclear Research, Geneva, Switzerland*
[8]*xLight, Palo Alto, California 94306, USA*
[9]*Cornell University, Ithaca, New York 14853, USA*
[10]*European Synchrotron Radiation Facility, Grenoble, France*
[11]*Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA*
[12]*Queen's University Belfast, Belfast, Northern Ireland, United Kingdom*
[13]*Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA*
[14]*Stanford University, Stanford, California 94305, USA*

(Received 9 December 2023; accepted 3 June 2024; published 6 August 2024)

Accelerator physics relies on numerical algorithms to solve optimization problems in online accelerator control and tasks such as experimental design and model calibration in simulations. The effectiveness of optimization algorithms in discovering ideal solutions for complex challenges with limited resources often determines the problem complexity these methods can address. The accelerator physics community has recognized the advantages of Bayesian optimization algorithms, which leverage statistical surrogate models of objective functions to effectively address complex optimization challenges, especially in the presence of noise during accelerator operation and in resource-intensive physics simulations. In this review article, we offer a conceptual overview of applying Bayesian optimization techniques toward solving optimization problems in accelerator physics. We begin by providing a straightforward explanation of the essential components that make up Bayesian optimization techniques. We then give an overview of current and previous work applying and modifying these techniques to solve accelerator physics challenges. Finally, we explore practical implementation strategies for Bayesian optimization algorithms to maximize their performance, enabling users to effectively address complex optimization challenges in real-time beam control and accelerator design.

DOI: 10.1103/PhysRevAccelBeams.27.084801

I. INTRODUCTION

Future accelerator-based experiments serving the high-energy physics, nuclear physics, and photon science communities will require a considerable increase in the capabilities of accelerator facilities to achieve the research aspirations of the next decade [1,2]. Higher energy and higher brightness particle beams with more stringent requirements on reproducibility will unavoidably require complex accelerator operation stemming from an increase in nonlinear phenomena, stringent beam parameter requirements, machine protection limits, and the varied needs of different user communities. Additionally, accelerator scientists designing future state-of-the-art accelerator facilities will need to explore and configure combinations of increasingly nonlinear and specialized accelerator elements to reach accelerator design goals, all while respecting practical constraints and minimizing construction costs.

*rroussel@slac.stanford.edu

Central to both of these challenges is the need to optimize a set of free parameters to attain a predefined objective. Examples of this include varying accelerator control parameters during operations to maximize performance (online tuning/optimization), identifying optimal parameters during the accelerator design process (offline simulated optimization), and matching simulated beam dynamics to experimental measurements (model calibration). Advancements in optimization algorithms enable us to tackle more challenging optimization problems (ones with more free parameters or more complex behaviors), which in turn, improves the performance and capabilities of accelerators.

Numerical optimization algorithms have long been used to address these challenges, but often suffer from slow convergence to optimal parameter sets, are unstable in noisy environments, and can get trapped in local limiting the complexity of optimization tasks that can be addressed in practice. Recently, a particular class of algorithms known as Bayesian optimization (BO) [3,4] has gained popularity inside the accelerator field as an efficient approach for solving both online and off-line optimization problems. These algorithms' inherent flexibility, low initialization effort, fast convergence, and robustness to noisy environments make them particularly useful for accelerator physics applications. Multiple groups inside the accelerator physics community have investigated the advantages and disadvantages of these algorithms for solving various accelerator physics problems. Furthermore, accelerator physics-specific modifications of basic BO components have been developed to leverage beam physics information, tailor optimization to practical operating challenges, and take advantage of high-performance computational clusters. With these developments, the study of BO techniques in the context of accelerator physics has matured to the point that these techniques are usable in regular accelerator operations and as a general high-performance optimization tool in simulation.

This review article aims to facilitate the wider use of BO techniques in accelerator physics by providing an easily accessible guide and reference for this class of optimization algorithms. We begin with a discussion of the optimization challenges faced by the accelerator physics community in regard to both online control of accelerator facilities and off-line optimization of simulations for accelerator design, which motivates the use of BO algorithms. We then discuss basic and advanced approaches to the principal components of BO algorithms: the Gaussian-process surrogate model most commonly used in BO; the definition of BO acquisition functions; and how the acquisition function is maximized to choose the next set of measurements. Throughout, we highlight how to incorporate beam physics information into BO algorithms in order to improve optimization performance. Finally, we conclude with a discussion that places BO in the context of other optimization algorithms, describes best practices for applying BO algorithms to solving optimization challenges, and future directions for research in this area.

II. BACKGROUND AND MOTIVATION

Optimization algorithms aim to solve the general problem

$$\mathbf{x}^* = \arg\ \max f(\mathbf{x}) \tag{1}$$

$$\text{s.t. } c_i(\mathbf{x}) \leq 0 \quad \forall\, i \in [1, \ldots, m]. \tag{2}$$

In the above formulation, Eq. (1) represents the objective function, wherein we seek a parameter set $\mathbf{x}^*$ that optimizes the function $f(\mathbf{x})$ subject to the m constraints specified in Eq. (2). These constraints may be bounds on the parameter set $\mathbf{x}$, or observables, such as safety and performance requirements. The formulation can be trivially transformed into a minimization problem by negating the objective function.

The difficulty of finding a solution to a generic optimization problem is influenced by the number of optimization parameters and the complexity of the objective and constraining functions. Increasing the number of optimization parameters exponentially increases the size of parameter space, often referred to as the *curse of dimensionality*. As a result, optimization algorithms that perform well when optimizing a small number of parameters (such as the fitting of three beam matrix elements to quadrupole scan data) can fail to find a solution in a reasonable amount of time when applied to higher dimensional problems (such as tuning the parameters of an entire accelerator beamline).

The complexity of the objective function also plays a role in the performance of optimization algorithms. Objective functions that are not convex have a number of local extrema, only one of which is the global optimum. In this case *local* optimization algorithms converge to local extremum near their initial starting conditions, while *global* optimization algorithms are designed to search the entire parameter space for a global extremum. For complex objective functions, finding the global optimum is often much more challenging [5].

A. Optimization challenges in accelerator physics

In addition to these general optimization challenges, online optimization of accelerators and off-line optimization of physics simulations add further, unique complications that need to be considered when selecting an ideal optimization algorithm.

1. Online accelerator control

Particle accelerators are challenging systems to optimize and control in practice, and potential optimization algorithms need to address these challenges. An illustration of some of these challenges is shown in Fig. 1. Typically, there are many possible settings that can be adjusted across multiple sub-systems to achieve the optimal beam

parameters. Measurements of beam qualities that serve as objective functions during optimization are often destructive and can be time-consuming, leading to losses in beam time available for experiments.

Accelerator measurements are also often subject to aleatoric (random noise) or epistemic (systematic) uncertainties. Random noise in accelerators makes it difficult for iterative algorithms to maintain stability throughout the optimization process. This noise can also change in amplitude as a function of accelerator parameters or changing environmental factors. The amplitude of noise can also be considered an optimization objective or constraint, given that it is often optimal to find solutions that lead to a relatively stable objective function value. Additionally, the limited resolution of accelerator diagnostics introduces systematic uncertainties in the objective function value. Finally, intermittent jumps (e.g., a spike in rf power, dip in beam charge, momentary fault from the machine protection system) in parameters need to be recognized and accounted for in automated optimization routines.

Particle accelerators are not stationary systems; they have time-dependent behavior on multiple timescales ranging from submilliseconds to hours. These behaviors include both expected time-dependent processes (such as slow loss of beam in a storage ring) or the combined effect of slow, unintended changes in the system (also known as "drift") that changes the relationship between settings and observed beam output. Drifts can come from many sources, such as changes in materials (e.g., loss of quantum efficiency in a photocathode) and the impact of daily and seasonal changes in temperature and humidity. Additionally, not all sources of drift are well characterized or measured.

Optimizing accelerator control parameters is often framed as a *multi-objective* problem, where the goal of optimization is to find a set of potential solutions that balance trade-offs between competing objectives. For example, many photoinjectors aim to simultaneously minimize both transverse beam emittances and bunch length of beams for high-brightness applications [6]. However, due to space charge effects, reducing the bunch length often increases the transverse beam emittance. Multiobjective optimization identifies a set of parameter configurations that provide ideal trade-offs between objectives, known as the Pareto front (PF). Once the PF has been identified, a single point on this PF can be selected based on objective preferences as a fixed operating point, or the entire front can be utilized to provide multiple operating modes for different applications.

Accelerators often operate in tightly constrained parameter spaces to limit beam losses that contribute to accelerator downtime, radiation generation, and hardware degradation. This is especially important for high-power beams, as even the lower-density edges of the beam distribution (or "halo") can damage equipment if the trajectory is not carefully controlled. These limits are often unknown prior to performing optimization, so algorithms need to learn valid and invalid regions of parameter space that satisfy the constraints on-the-fly during optimization. On the other hand, there are cases where operational constraints are not as strict, such as beam losses in lower power facilities or nonsafety related beam quality constraints (maximum beam emittance or energy spread). In these cases, occasional violations of the constraints can be tolerated if they lead to increased convergence speed to optimal values. Algorithm design for online accelerator operations needs to balance conservative adjustments of accelerator parameters to avoid constraint violations with the need to explore the input space to find optimal solutions.

The type of operating conditions for a particular accelerator also impacts how challenging it is to arrive at an optimal configuration. Some particle accelerators deliver highly customized beams to their users, which requires a new combination of accelerator settings for each request. Large changes in machine setup introduce additional challenges, such as the need to deal with path-dependent processes like magnetic hysteresis and mechanical backlash. Additionally, rapid changes to accelerator parameters can lead to instabilities in the machine due to interacting feedback algorithms in multiple accelerator subsystems. The degree to which these processes need to be considered depends in part on whether the accelerator must undergo somewhat global optimization frequently, as opposed to keeping a single configuration stable for long periods of time.

2. Simulation-based optimization of accelerator systems

Optimization algorithms are also used in simulation to design new accelerator components and facilities, as well as calibrate physics models to experimental measurements. Simulated optimization shares some of the same challenges as online optimization, including satisfying constraints, limited evaluations, and balancing the trade-offs between multiple competing objectives.

Detailed physics simulations are often used in the design of new particle accelerators and new experimental setups. To include the full detail of nonlinear beam dynamics or collective effects, computationally intensive, high-fidelity particle-in-cell simulations are used to accurately make predictions of real-world beam dynamics. However, running these simulations consumes a significant amount of computational resources and run time. Thus, using algorithms that reduce the need for high-fidelity simulations is essential to keep computational costs at a minimum.

To speed up optimization, multiple simulations can be performed in parallel on high-performance computing clusters. Additionally, low-cost, approximate simulations can be used to perform optimization, albeit with less predictive accuracy. In an ideal scenario, multiple, inexpensive evaluations of an approximate model would be used to identify promising regions of parameter space before evaluating expensive, high-fidelity simulations using those parameters. Accelerator scientists have typically implemented this strategy manually by preselecting a suitable balance between

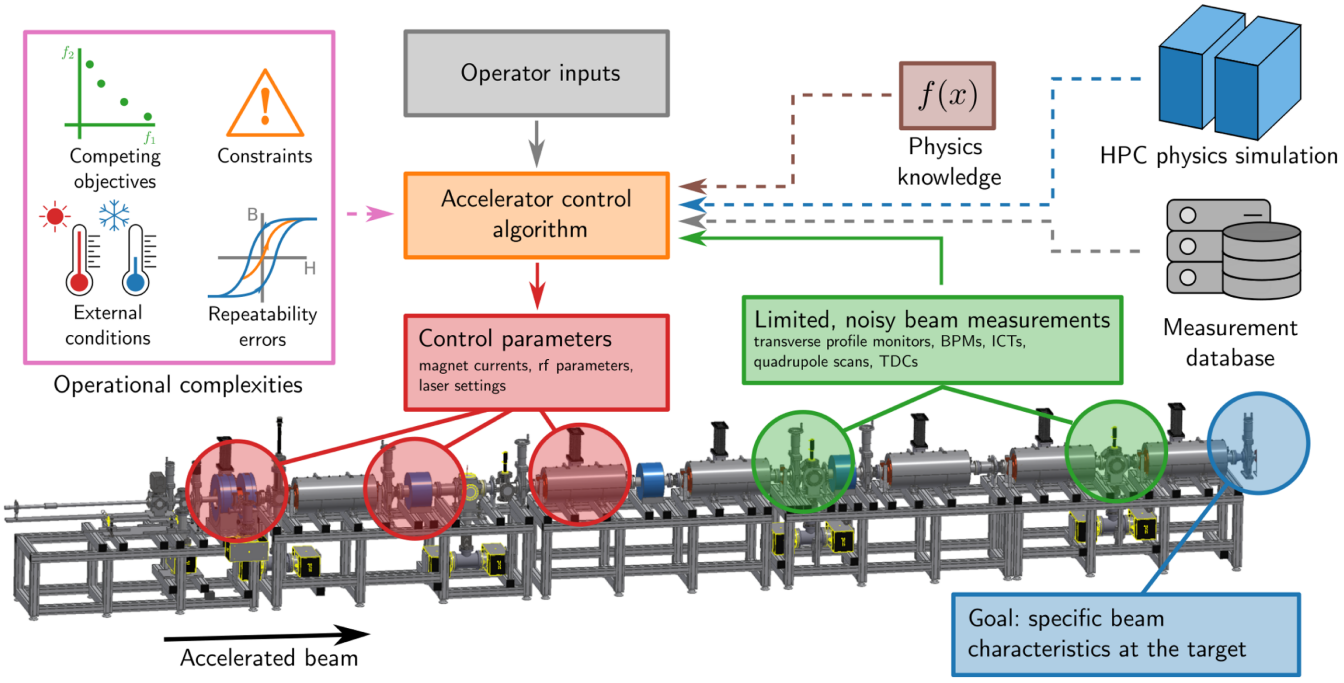

FIG. 1. Overview of challenges in using optimization algorithms for online accelerator control. Accelerator control algorithms make decisions about setting a wide variety of accelerator parameters in order to control beam parameters at target locations. Optimal decision making takes into account limited online accelerator measurements, as well as various sources of prior knowledge about the accelerator, including previous measurements, physics simulations, and physics principals. Optimization must also consider complicated aspects of realistic accelerator operation including external conditions, feedback systems, safety constraints, and repeatability errors.

simulation precision and computational cost. However, recent advances in optimization algorithms enable an automated approach to this process, where simulations of different fidelity can be combined into a single optimization to reduce the overall run time and computational cost, as shown in Fig. 2.

B. Optimization algorithm selection

A variety of different algorithms has been developed to find solutions to the optimization problem described in Eq. (1). One class of algorithms, known as *Iterative* optimization algorithms, is a popular choice. Given an initial point in parameter space, these algorithms generate a point or set of points that are evaluated using the objective and constraining functions. Results from the evaluations are then passed back to the algorithm to generate the next point(s) to be evaluated. The final solution is determined once the algorithm reaches a termination condition, for example, a fixed number of iterations or a satisfactory objective function value.

Selecting the right algorithm for a given optimization task is critical to success, as it directly influences the quality of the final solution and resources needed to execute the applied routine (e.g., required beam time and computational resources). To compare different optimization algorithms, we define the following terms: The costs of evaluating the objective/constraining functions, for example, beam time at an accelerator, computational assets at a computing cluster, personnel time resources, or financial expenditure, is referred to as the *evaluation cost*. We refer to the number of objective/constraint evaluations needed by a given algorithm to reach a predefined objective target of a particular optimization problem as the algorithms' *sample efficiency* or *convergence speed*. Additionally, different optimization algorithms can perform a variety of computations to generate the next point(s) to be evaluated. The computational costs of performing these calculations are referred to here as the *decision-making cost*. Finally, in certain cases, algorithms may utilize or require upfront effort before they can be used to solve optimization problems, such as historical data gathering or pretraining of decision-making agents, referred to here as *initialization costs*.

Choosing the correct algorithm for solving an optimization problem requires balancing the trade-offs between the different costs associated with performing optimization. For example, if the evaluation cost of the problem is high, it is advantageous to use algorithms that are more sample efficient, even if they require high decision-making costs relative to other algorithms (provided that the decision-making cost is low relative to the evaluation cost). On the other hand, if evaluation costs are low relative to the decision-making costs of a given algorithm, costs associated with decision making will be dominant over other costs. In this instance, it makes sense to use algorithms that have low decision-making costs, even if they have poorer sample efficiency resulting in more evaluations of the objective.

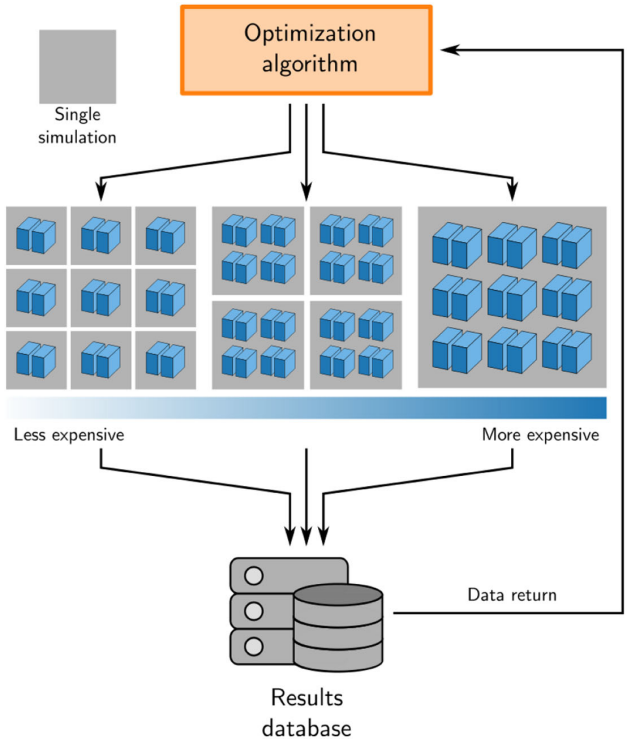

FIG. 2. Overview of optimization challenges in accelerator physics simulations. Ideal algorithms aim to minimize the computational cost of performing optimization by orchestrating parallel simulation evaluations at multiple fidelities ranging from analytical models to high-fidelity (computationally expensive) simulations. Correlations between simulation predictions at different fidelities can be leveraged to reduce the number of high-fidelity simulation evaluations needed to find an ideal solution at the highest-fidelity level.

However, when optimizing problems that contain safety related constraints that protect equipment or personnel, higher decision-making costs may be necessary to obey constraints during optimization. Finally, incurring higher initialization costs, such as conducting prior measurements or performing precomputations, can be used by the optimization algorithm to increase sample efficiency. This can be especially advantageous if the same optimization problem (with minor perturbations to the objective function) needs to be solved repeatedly, such as is the case of using the same algorithm in day-to-day operations, because initialization costs are incurred only once. Effectively balancing these costs is key to selecting the best algorithm for a given optimization task.

C. Bayesian optimization

Bayesian optimization (BO) is an iterative, model-based optimization algorithm that is particularly well suited for sample-efficient optimization of noisy, expensive-to-evaluate objectives. Current examples of problems where BO is effective include tuning accelerator parameters, such as magnet power supplies (which can take several seconds to adjust), optimizing objectives that are time consuming to measure (such as transverse beam emittance), or in simulated contexts where physics simulations require significant computational resources (computing time and processing power) to make predictions. On the other hand, BO algorithms are generally not well suited for fast-feedback applications (at or below the 1 Hz level) such as low-level rf control, although BO algorithms can be suitable for optimizing the hyperparameters of fast-feedback controllers [7].

Similar to other optimization algorithms, BO algorithms require at least one initial point to start making decisions about future points to observe. Initial points are often selected using randomly sampled points in a local region of parameter space or preselected points known to be near the objective extremum. Alternatively, it is also possible to include data directly from historical datasets or previous optimization runs. Using a large set of historical data to initialize BO is often referred to as a *warm start* while initializing BO with a small number of input points is referred to as running BO *from scratch*.

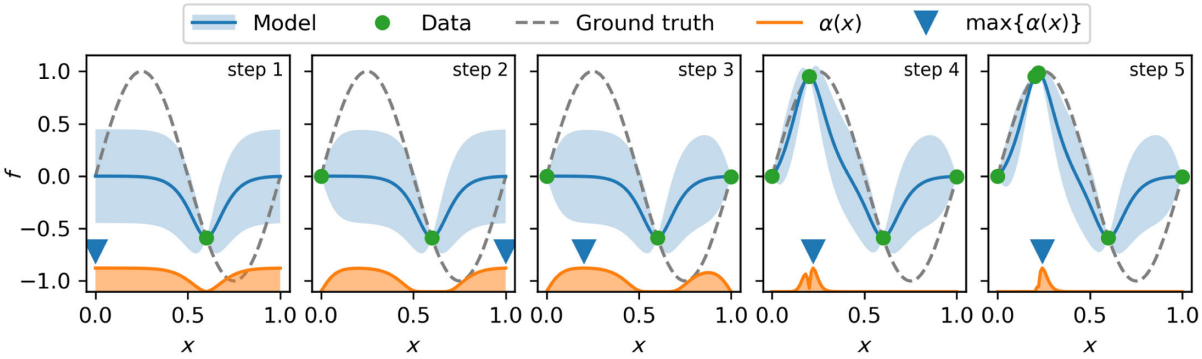

FIG. 3. Illustration of the Bayesian optimization process to find the maximum of a simple function. A Gaussian process (GP) model makes predictions of the function value (solid blue line) along with associated uncertainties (blue shading) based on previously collected data. An acquisition function then uses the GP model to predict the "value" of making potential future measurements, balancing both exploration and exploitation. The next observation is chosen by maximizing the acquisition function in parameter space. This process is repeated iteratively until optimization goals have been reached.

ALGORITHM 1. Bayesian optimization.

Require: Objective function f, observation dataset $\mathcal{D}_{\mathcal{N}}$, GP prior $M = \mathcal{GP}[\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')]$, acquisition function $\alpha(\mathbf{x})$.
1: **for** $t = 1, 2, \ldots$ **do**
2: Decide a new sample point $\mathbf{x}_{\text{new}} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x})$.
3: Query the objective $y_{\text{new}} = f(\mathbf{x}_{\text{new}}) + \epsilon$.
4: Update $\mathcal{D}_{\mathcal{N}}$ and the GP model.
5: **end for**

In general, BO consists of three steps, as illustrated in Fig. 3, and is summarized in Algorithm 1. The first is the construction of a statistical surrogate model of the objective and constraining functions based on measured data, often using Gaussian process (GP) modeling [8]. The second step is the definition of an acquisition function based on the GP model, which defines the relative "value" of potential future measurements in input space in order to achieve optimization goals. The final step solves for the point (or set of points) that maximizes the acquisition function and is thus predicted to provide the most value toward optimization goals. Points that are selected in the last step are then passed to the objective and constraint function(s) to be evaluated; the results of which are then passed back to the algorithm to be incorporated into the model dataset. This process repeats until an optimization criterion is met.

An additional benefit of BO is that the model created and trained during the optimization process can also be used outside of the context of optimization. For example, the model can provide information about objective function sensitivities to accelerator parameters, be integrated as a fast-executing surrogate into other models of the accelerator, or be used to identify unknown parameters of the beamline, such as element misalignments or hysteresis effects. As a result of the BO sampling process, these models are often most accurate in regions of parameter space that are of the highest interest, namely regions of parameter space that are near optimal parameter sets.

D. Demonstrations of BO in accelerator physics

Bayesian optimization has already been used to solve a wide variety of optimization problems in accelerator physics. These demonstrations include (i) single-objective, online, and off-line optimization of accelerator parameters, e.g., of magnetic optics, rf parameters, in conventional linear [9–15] and circular [16,17] accelerators, as well as novel accelerator concepts [18–23], (ii) online optimization that leverages prior physics knowledge or simulations [10,24,25] (Secs. III C 1 and III C 2), (iii) time-dependent optimization to maintain optimal tuning configurations in problems subject to drift [11,26,27] (Sec. III C 4), (iv) online optimization subject to repeatability errors (hysteresis and motor backlash) [28] (Sec. III C 6), (v) autonomous characterization of objective functions [29] (Sec. IV B 1), (vi) optimization with unknown constraints [30–32] (Sec. IV B 2), (vii) multiobjective optimization to discover ideal trade-offs between competing objectives in experiments [22,33] and simulations [34–36] (Sec. IV B 3), (viii) Bayesian algorithmic execution, e.g., optimization of beam emittance using virtual quadrupole scans [37] (Sec. IV B 4), and (ix) multifidelity optimization, e.g., of beam dynamics and plasma wakefields in simulations [35,38] (Sec. IV B 6).

In the following sections, we describe BO techniques in detail; first, by introducing common approaches and methods for each step. We then describe advanced modifications of basic techniques that have been shown to be advantageous toward solving accelerator physics problems.

III. GAUSSIAN PROCESS MODELING

Bayesian optimization uses a computational surrogate model of the objective function in order to inform the selection of new measurement points in input space. In practice, the surrogate model should use data collected during optimization to make predictions of the objective function value as well as provide an estimate of corresponding uncertainties with those predictions. While any surrogate model with these properties could potentially be used in this context, models known as "Gaussian Processes" (GPs) [8] are often used.

A. Bayesian inference

Before starting a discussion of models used in BO, it makes sense to first develop a conceptual understanding of *Bayesian statistics*. A Bayesian interpretation of probability expresses a degree of belief in an event, or a probability distribution, based on prior knowledge of that event. This is different than a frequentest view of probability that reflects the measured outcomes of many trials. Bayesian statistics uses *Bayes' rule* to predict the conditional likelihood of an event A occurring given an event B happened as

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}, \tag{3}$$

where $p(B|A)$ is known as the *likelihood function*, $p(A)$ is the *prior* probability distribution, $p(B)$ is the *marginal likelihood* or the *evidence*, and $p(A|B)$ is the *posterior* probability.

To make the interpretation of Bayes' rule more concrete, we examine fitting a linear model $f(x) = w_0 x + w_1$ to experimental measurements corrupted by noise $y = f(x) + \epsilon$ as is shown in Fig. 4. The goal of this analysis is to determine the likelihood of the two model parameters $\{w_0, w_1\}$ given a collection of experimental measurements $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\}$. Using Bayes' rule, the posterior probability distribution of these parameters is given by

$$p(w_0, w_1|\mathcal{D}) = \frac{p(\mathcal{D}|w_0, w_1)p(w_0, w_1)}{p(\mathcal{D})}. \tag{4}$$

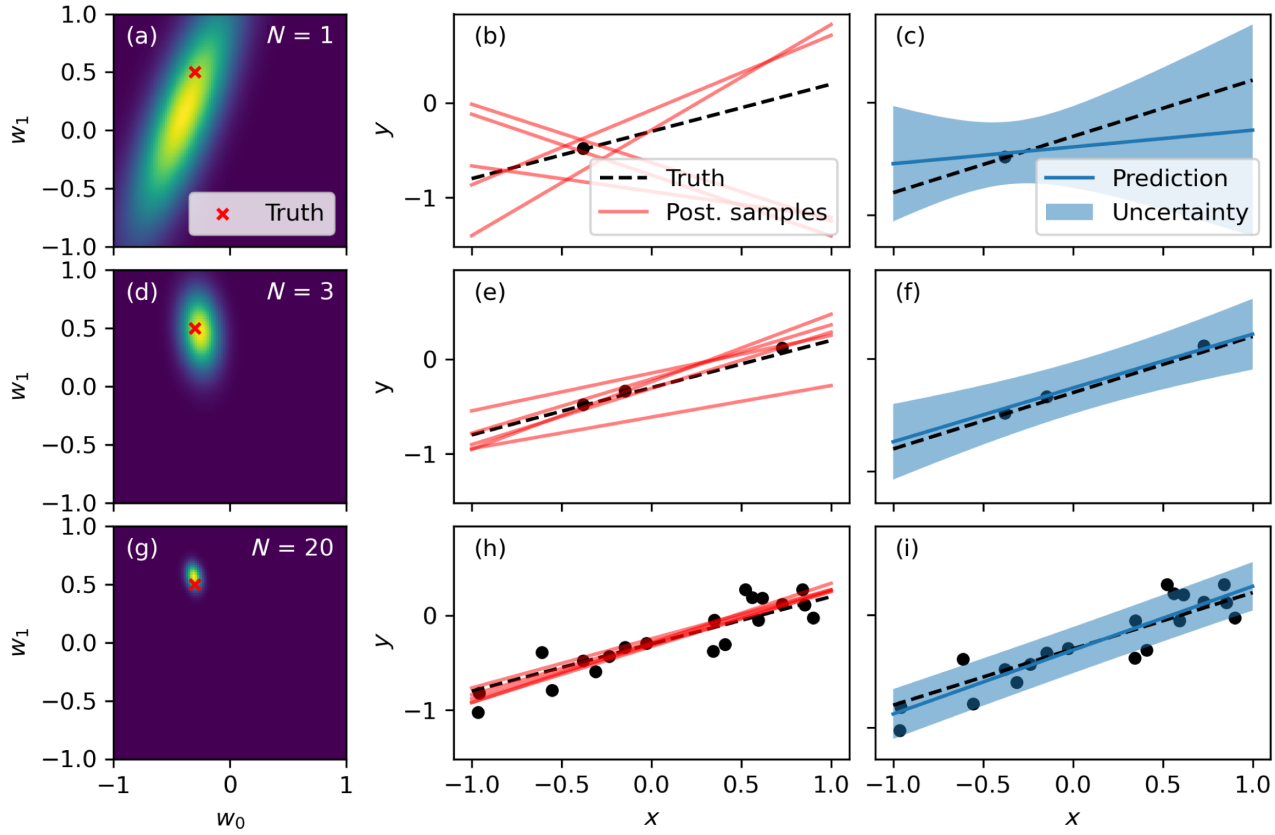

FIG. 4. Illustration of Bayesian regression using a linear model $f(x) = w_1 x + w_0$. (a,d,g) Posterior probability density of the linear weights $\{w_1, w_0\}$ conditioned on N observations of the function $y = f(x) + \epsilon$. (b,e,h) Model predictions using random samples of $\{w_1, w_0\}$ drawn from the posterior probability distribution. (c,f,i) Predictive mean (solid line) and 90% uncertainty intervals (shading) of the posterior model. Red cross and black dashes denote true parameters and values of the function $f(x)$, respectively. Reproduced with permission from [39].

where the likelihood $p(\mathcal{D}|w_0, w_1)$ captures how well the linear model with the given parameter values $\{w_0, w_1\}$ represents the measured data, and $p(w_0, w_1)$ represents the prior probability distribution of the weights. In this case, the prior distribution of the weights is a multivariate Gaussian distribution centered at the origin. This prior distribution is equivalent to adding a regularization term to least-squares curve fitting, which aims to prevent overfitting by penalizing large parameter values.

After a single observation, the posterior probability distribution of the weights is shown in Fig. 4(a). Based on the single measured data point, Bayes' rule predicts a positive correlation between the y intercept (w_0) and the slope (w_1). This is evident in function samples drawn from the posterior distribution shown in Fig. 4(b). These samples can be collected into a distribution that predicts the mean value of the function and associated uncertainty as is shown in Fig. 4(c).

As additional measurements are introduced into the model, Figs. 4(d)–4(i), the likelihood and posterior probability distributions become sharper as a smaller range of parameters leads to accurate models of the experimental data. How rapidly the posterior probability distribution shrinks according to new evidence depends on the relative "strengths" of the prior and likelihood distributions. A strong prior probability distribution on the weights (one that is highly peaked in a small local region) will result in a similar posterior distribution unless significant experimental evidence that is contrary to the prior is incorporated into Bayes' rule via the likelihood. On the other hand, if the prior distribution is relatively weak (i.e., nearly uniform across parameter space), then it has relatively little impact on the posterior distribution.

The process of determining the posterior probability distribution of model parameters based on observed data and Bayes' rule is referred to as *Bayesian inference*. In most cases, determining the exact posterior probability distribution for the entire parameter space requires performing integrals that are computationally intractable to compute, specifically when evaluating the evidence term in Bayes' rule $p(\mathcal{D})$. Rather than directly assessing the posterior probability distribution, a variety of alternative analytical techniques are used to perform inference. First is *maximum likelihood estimation* (MLE), which estimates pointlike values of the model parameters θ by solving for the point θ^* that maximizes the likelihood term (which ignores any

priors on the parameters)

$$\theta^*_{\mathrm{MLE}} = \arg\max_\theta p(\mathcal{D}|\theta). \tag{5}$$

If the likelihood takes the form of a normal distribution, this is equivalent to performing mean squared error curve fitting.

The second analysis method is *maximum a posteriori* (MAP), which also determines pointlike values of the parameters θ, this time by maximizing a quantity that is the mode of the posterior distribution

$$\theta^*_{\mathrm{MAP}} = \arg\max_\theta p(\mathcal{D}|\theta)p(\theta), \tag{6}$$

which incorporates the prior without having to compute the full posterior probability distribution. Finally, while it is possible to determine the full posterior probability distribution of θ [40], it is often analytically intractable or computationally expensive to evaluate and is usually not required in the context of BO.

B. Gaussian process modeling basics

Gaussian process models [8] are nonparametric models that use Bayes' rule to model unknown functions by leveraging correlations between function values at different locations in parameter space. As opposed to Bayesian regression, which uses Bayes' rule to identify probability distributions of model parameters, GP models use Bayes' rule to predict probability distributions of function values at arbitrary locations in parameter space using measured data. An additional way to view this distinction is that parametric models simply rely on the value of model to make predictions, while nonparametric models, such as GPs, explicitly rely on training datasets to make predictions.

We start by assuming that the output y of a function f at input parameter $\mathbf{x}$ is given by

$$y = f(\mathbf{x}) + \epsilon, \tag{7}$$

where random corrupting noise is drawn from ($\sim$) a normal distribution ($\mathcal{N}$), $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ with a variance σ_ϵ^2. The Gaussian process is defined as

$$f(\mathbf{x}) \sim \mathcal{GP}[m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')], \tag{8}$$

where $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ is referred to as the *prior mean function*, and $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[\{f(\mathbf{x}) - m(\mathbf{x})\}\{f(\mathbf{x}') - m(\mathbf{x}')\}]$ is commonly called the *covariance function* or *kernel*. The probability distribution of the observable y is given by our assumed *likelihood*, which in this case is a normal distribution $p(y|f(\mathbf{x}), \sigma_\varepsilon) = \mathcal{N}(f(\mathbf{x}), \sigma_\varepsilon^2)$. To simplify calculations, the prior mean function is often specified to be $m(\mathbf{x}) = 0$, although a fixed nonzero prior mean can also be learned from the data.

Given a set of n collected data samples $\mathcal{D} = \{X, \mathbf{y}\}$, we can make predictions for the probability distribution of the function value evaluated at n_* test points using Bayesian inference. The resulting posterior distribution $p(\mathbf{y}_*|X_*, \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\sigma}_*^2)$ with the mean and variance given by [41]

$$\boldsymbol{\mu}_* = K(X_*, X)[K(X, X) + \sigma_\epsilon^2 I]^{-1}\mathbf{y}, \tag{9}$$

$$\boldsymbol{\sigma}_*{}^2 = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_\epsilon^2 I]^{-1}K(X_*, X)^T, \tag{10}$$

where $K(X, X)$ is an $n \times n$ covariance matrix between each dataset element locations, $K(X_*, X)$ is an $n_* \times n$ covariance matrix between test points and dataset element locations, $K(X_*, X_*)$ is an $n_* \times n_*$ covariance matrix between test point locations, and I is the identity matrix.

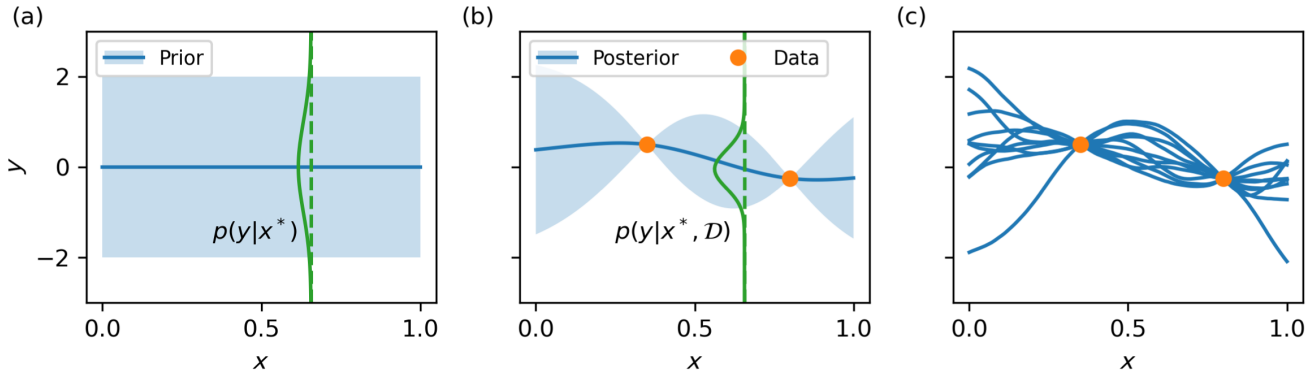

FIG. 5. Illustration of GP model predictions. (a) Prior model prediction of the function mean (solid blue line) and confidence interval (blue shading) at a set of test points in parameter space. The probability of the output value y at any given test point x^* is a normal distribution. (b) The posterior GP model also predicts normal probability distributions at each test point, conditioned on the dataset $\mathcal{D}$. (c) Individual function samples can also be drawn from the posterior GP model and can be used for Monte Carlo computations of function quantities.

An example of GP predictions is shown in Fig. 5, assuming that the noise parameter $\sigma_\epsilon = 0$. Figure 5(a) shows the prior mean and confidence bounds (equal to 2σ above and below the mean) of the observable y for a set of 100 test points in the domain $x^* \in [0, 1]$. At an arbitrary point in parameter space, the GP prior distribution $p(y|x^*)$ is a normal distribution with a mean of zero and a unit variance. By adding a dataset $\mathcal{D}$ to the GP, the model predictions are updated to form the posterior distribution as shown in Fig. 5(b). Posterior predictions at a single test point also take the form of normal distributions with means and variances conditioned on the dataset according to Eqs. (9) and (10). We can also draw individual function samples at points in parameter space from the posterior distribution, as shown in Fig. 5(c). These function samples are generated by drawing multiple random values from the normal distribution at every point in input space.

Conceptually, GP models use Bayes' rule to derive a posterior probability distribution of the function value $f(\mathbf{x})$ conditioned on the observed dataset and covariances in function values between observed data and test points. These covariances are defined by the kernel function $k(\mathbf{x}, \mathbf{x}')$ and a likelihood function (which describes probabilities due to measurement noise). A physical analog of GP modeling is a vibrating string with a collection of fixed nodes along the string length. The possible locations of the string at any point along its length are constrained by where the nodes are located on the x-y plane (observed data) and the elasticity of the string (kernel function). For a given string, we can be quite confident where the string is in space close to fixed nodes. However, far away from any nodes, the string position possibilities can vary widely. Increases in the elasticity of the string creates additional uncertainty in both of these cases owing to its ability to stretch, which corresponds to weaker covariances between function values.

1. Kernel function definition

By defining the covariances of function values between different locations in parameter space, the kernel function dictates the overall functional behavior of the predictive model. The selection of a particular kernel function is usually based on prior knowledge of the real function's behavior in parameter space and is critical to creating accurate models with limited amounts of data. Kernel functions often contain *hyperparameters*, which alter the high-level functional behavior of the GP posterior and can be specified prior to modeling or inferred from training data. Kernels are generally divided into two categories: stationary and nonstationary.

Stationary kernels are invariant under translations in input space $k(\mathbf{x}, \mathbf{x}') = k_S(||\mathbf{x} - \mathbf{x}'||)$, which means it only depends on the relative positions of its two inputs $\mathbf{x}$ and $\mathbf{x}$', and not on their absolute positions. This feature makes stationary kernels a popular choice for modeling arbitrary functions when limited prior information is present.

One of the most basic stationary kernels is the Radial Basis Function kernel (RBF). The RBF kernel is defined as:

$$k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right), \tag{11}$$

where l is the length scale hyperparameter of the kernel. While the RBF kernel's simplicity makes it easy to use and adapt to specific purposes (see Sec. III C 1), it results in predictions that are infinitely differentiable, which are generally too smooth for describing realistic functions.

A more generalized version of the RBF kernel is the Matérn kernel. The Matérn kernel is defined as

$$k_{\mathrm{MA}}(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{d}{l}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{d}{l}\right). \tag{12}$$

Here, $d = ||\mathbf{x} - \mathbf{x}'||$ represents the Euclidean distance between inputs, Γ is the gamma function, and K_ν is the modified Bessel function of the second kind. The length scale of the kernel is denoted by l, and ν controls the smoothness of the resulting function. As $\nu \to \infty$, the Matérn kernel converges to the RBF kernel.

Commonly used values for ν are $\nu = 1.5$ for once differentiable functions and $\nu = 2.5$ for twice differentiable functions. Limiting the differentiability enables GP models with Matérn kernels to accurately predict realistic physical processes. As a result, the Matérn kernel with $\nu = 2.5$ is often employed as a starting point for modeling physical functions in the absence of prior information.

For modeling functions that are expected to be periodic, a periodic kernel can be used to increase model accuracy for small datasets. A periodic kernel, also called a Exp-Sine-Squared kernel, is defined as

$$k_{\mathrm{PER}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{2\sin^2(||\mathbf{x} - \mathbf{x}'||/p)}{l^2}\right), \tag{13}$$

where l is the length scale of the kernel and p is the periodicity of the kernel.

Hyperparameters of these kernels control high-level model behavior by modifying the covariance between function values at different points in parameter space. For example, Fig. 6 shows how the length scale hyperparameter of a stationary kernel affects GP predictions. Models that contain kernel functions with short length scales vary rapidly to precisely match the training data, which leads to overfitting. As the length scale increases, the smoothness of the model prediction increases, capturing more of the general trend of the training data without overfitting. However, increasing the length scale beyond this point leads to diminished model accuracy. Selecting hyperparameter values depends on improving the accuracy of the GP model while reducing the complexity of the model, thus preventing overfitting of the data (see the next section for a detailed discussion).

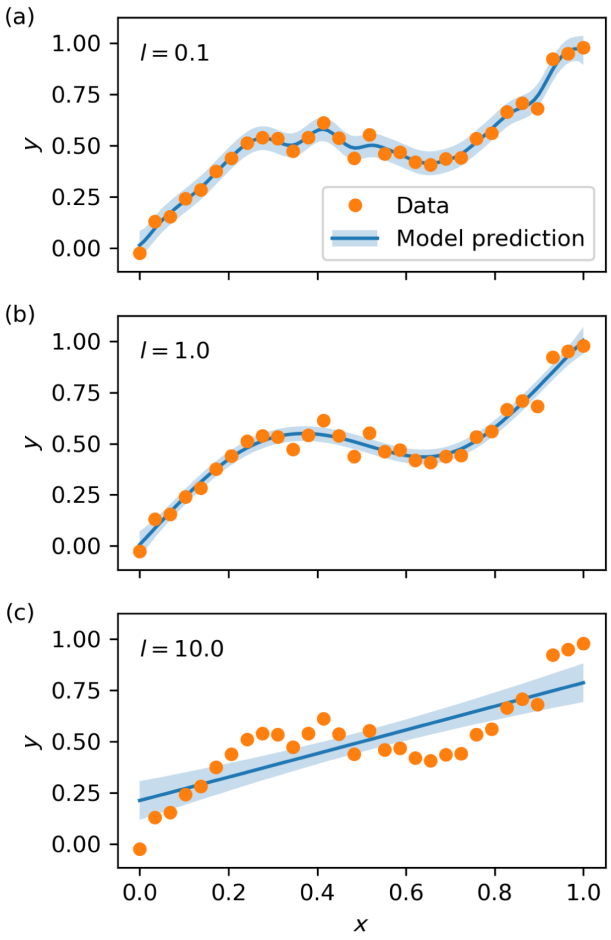

FIG. 6. Visualization of how the length scale hyperparameter l effects GP modeling. Three GP models are trained on the same dataset using a Matérn kernel with fixed length scales of (a) 0.1, (b) 1, and (c) 10. Remaining hyperparameters are trained by maximizing the marginal log-likelihood.

One straightforward modification of stationary kernels often used in practice is replacing the scalar length scale hyperparameter (which would be isotropic) with a vector of independent length scales corresponding to each optimization parameter, creating an anisotropic kernel. In this case, the RBF kernel, for example, can be specified by

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T M(\mathbf{x} - \mathbf{x}')\right) \quad (14)$$

with $M = \text{diag}(\mathbf{l})^{-2}$ where $\mathbf{l}$ is a vector of positive real values. This technique is often referred to as *automatic relevance determination* [42] and can be used to identify the sensitivity of the objective function to each optimization parameter. A long length scale implies weak dependence of the objective function on a particular parameter while a small length scale implies strong dependence.

The flexibility of this approach can be expanded even further by specifying a full positive semidefinite matrix $M = \Lambda^T\Lambda + \text{diag}(\mathbf{l})^{-2}$, where Λ is an upper triangular matrix, often referred to as *factor analysis distance* due to the analogy with factor analysis methods used to find low-rank decomposition of the data along arbitrary axes in parameter space. This parameterization is used less often in practice due to the large datasets necessary to learn the covariances on the fly during optimization. It can, however, be useful in cases where the decomposition can be determined prior to conducting optimization and the low-rank behavior is not aligned with individual parameter axes, for example, in cases where quadrupole parameters are tuned (see Sec. III C 1).

Automatic relevance determination and factor analysis distance do increase the numerical complexity of calculating the GP kernel when compared to isotropic kernels with a single length scale. This in turn, adds additional decision-making costs to the BO algorithm. However, provided that enough information is present to learn or specify *a priori* the underlying low dimensional structure of the function, automatic relevance determination and factor analysis distance can significantly improve the accuracy of GP models in high dimensional parameter spaces with fewer data points, leading to higher sample efficiency.

Nonstationary kernels depend explicitly on the locations of the two inputs $\mathbf{x}$ and $\mathbf{x}'$. Using basic nonstationary kernels can provide more accurate predictions with fewer data points, at the cost of reduced model flexibility. A commonly used nonstationary kernel is the polynomial kernel [43]. A polynomial kernel of degree p is defined as

$$k_{\text{POL}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^p, \quad (15)$$

where $c \geq 0$ is a constant offset parameter. Using a polynomial kernel in a GP model is equivalent to performing Bayesian regression of data using the same-order polynomial. Functional samples drawn from the GP model are then also polynomial functions of the same order as the kernel.

2. Determining model hyperparameters

The hyperparameters of the GP kernel can be learned from training data gathered during optimization or specified *a priori* using prior knowledge of the objective function. A common strategy for learning the hyperparameters from experimental data is maximizing the *marginal log-likelihood* (MLL) of the GP model with respect to the hyperparameter values. While in most cases calculating the marginal likelihood requires performing analytically intractable integrals, the marginal likelihood of GP models with Gaussian likelihoods can be calculated analytically and is given by

$$\log\, p(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi, \quad (16)$$

where θ is the set of GP model hyperparameters contained in $K_y = K(X, X) + \sigma_\epsilon^2 I$, and n is the number of

training samples. The MLL has three terms, each having an interpretable role. The first term, which is the only term that contains training data, is the data fit term which is maximized when model predictions accurately predict experimental data. The second term describes model complexity and is maximized given the simplest model, i.e., models whose kernel matrices have determinants close to zero. The final term is a normalization constant based on the number of training points in the dataset. Maximizing the MLL naturally regularizes the fitting of the GP, resulting in model hyperparameters that create the simplest model that accurately reproduces the training data. For relatively small datasets (<300 data samples), maximizing the MLL takes a few seconds on most modern CPUs, making it feasible to perform this process during each iteration of BO (see Sec. VI F for details).

Alternatively, fixed individual hyperparameter values can be specified before modeling occurs, based on prior knowledge of the function, either from previous sets of data or physics knowledge. While fixing hyperparameter values circumvents the need for retraining the model at each optimization step during BO, this limits the ability of BO to adapt to novel functional behavior that is not well characterized by the fixed hyperparameter values.

As maximizing the MLL is itself an optimization problem, this process suffers from the same complexities and challenges associated with solving general optimization problems in practice. A wide variety of numerical optimization algorithms can be used for this purpose, given that the number of hyperparameters that are included inside the GP model is generally small (<5–10). Current state-of-the-art software packages developed for GP modeling (see Sec. VI E) employ two strategies to maximize speed and robustness when optimizing the MLL.

The first strategy is to use so-called *differentiable* calculations [44], which allow cheap computation of the MLL gradient with respect to the hyperparameters. This enables the use of gradient-based optimization algorithms that scale well toward performing optimization given a large number of hyperparameters. As gradient descent optimization algorithms often converge to local extrema, optimization can be repeated in parallel, starting with randomly chosen initial points in hyperparameter space to improve the chances of finding a global extremum.

The second strategy used to improve MLL maximization robustness is training data normalization and standardization. As is common in other machine learning disciplines, it is recommended that training datasets are transformed such that they are near unity value when passed to the model, thus preserving unit scale gradients with respect to hyperparameters. For GP modeling, it is common to normalize input data into the unit domain [0, 1] and standardize the outcome data such that it has a mean of 0 and a standard deviation of 1 (to match the default zero prior mean and unit standard deviation in most GP modeling frameworks).

These two strategies make maximizing the MLL fairly robust in practice, such that monitoring and customizing the fitting of model hyperparameters in BO is only necessary in specialized cases and model debugging.

3. Observation noise and heteroskedasticity

In most cases when performing online optimization of accelerator parameters, measurements of the objective function are corrupted by noise and/or systematic uncertainties. Through the use of Bayesian inference, GP models explicitly support a notion of measurement uncertainty when making predictions. Depending on the application, GP models can be tailored to account for measurement uncertainty in a variety of ways based on measurements or prior physics information.

The most straightforward method for representing measurement uncertainty uses a noise hyperparameter σ_ϵ that is incorporated into Eqs. (9) and (10) by assuming a fixed Gaussian model of the uncertainty for all measurements. This *homoskedastic* uncertainty assumption adds σ_ϵ^2 terms to the diagonal elements of the kernel matrix, in what is sometimes referred to as Tikhonov regularization or ridge regression [45]. In cases where no noise is present, as in deterministic simulations, this parameter can be set to zero, resulting in GP models that make predictions which exactly match the training data, as shown in Fig. 7(a). In the case of experimental measurements containing noise, the noise hyperparameter can be determined during optimization alongside other model hyperparameters by maximizing the MLL. This process serves to regularize the GP model, mitigating high-frequency behavior and treating it as uncertainty at measurement locations, as exemplified in Fig. 7(b).

In some situations, measurement uncertainty is not constant throughout parameter space. This may be the result of varying systemic uncertainties in the measurement or varying levels of stochastic noise in different portions of parameter space. In this case, model uncertainty can be different for each measurement or *heteroskedatic* in nature. If the observation uncertainty can be estimated, e.g., by taking repeat measurements to estimate stochastic noise, and/or by specifying systematic measurement uncertainty, this information can be included for each point individually in a heteroskedastic model. In this case, different values of $\sigma_{\epsilon,i}^2$ can be added to the diagonal elements of the covariance matrix for each data point y_i. This allows for individual measurement uncertainty to be accounted for explicitly in the GP model, as illustrated in Fig. 7(c). An alternative approach is to use a second GP (or any other deterministic model) to describe the variance as a function of input parameters.

4. Computational costs

If the likelihood of the GP model is a normal distribution, then calculating the posterior distribution is analytical via matrix computations. However, for more complex models,

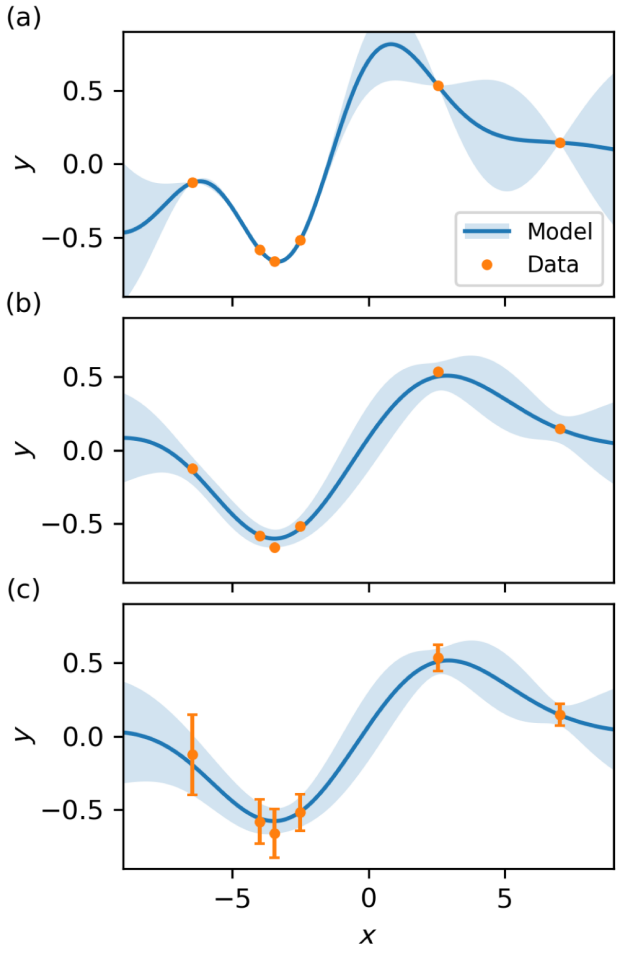

FIG. 7. Examples of GP modeling with varying treatment of measurement noise. (a) Shows a GP model containing zero noise, forcing the GP prediction to fit experimental data exactly. (b) Shows a GP model trained on the same data with a fixed (homoskedastic) noise parameter. (c) Illustrates a GP model incorporating heteroskedastic noise, where the data variance for each point is explicitly specified.

the posterior cannot be obtained analytically and may require the use of a sampling algorithm such as Markov Chain Monte Carlo (MCMC) [46] to estimate the posterior, which is known to be more computationally intensive.

Calculation of the GP posterior can become a significant bottleneck given a large dataset of training points. The computational cost of evaluating GP model predictions is primarily due to the matrix inversion operations in Eqs. (9) and (10) which has a computational cost of $O(n^3)$ where n is the number of training points (note that it is independent of the dimensionality of $\mathbf{x}$). As a result, the decision-making cost of BO algorithms can increase substantially as the number of optimization iterations increases due to the need to train and evaluate GP models (benchmarking of these computational costs on modern hardware architectures can be found in Sec. VI F). Thus, when using BO, it is advantageous to find strategies that reduce the number of training data points needed to make an accurate model of the objective function that informs optimization. This is where advanced modeling techniques come into play.

C. Advanced Gaussian process modeling techniques

One of the primary goals of advanced modeling techniques is to encode prior information into the GP model such that it makes more accurate predictions with smaller sets of data. This prior information can come from a number of different sources, including (but not limited to) physics knowledge, historical datasets, and/or prior optimization runs. Improving the predictive accuracy of GP models prior to starting optimization improves the quality of decisions made by BO, reducing the average number of iterations needed to reach convergence. Furthermore, reducing the number of data points needed for accurate modeling reduces the computational cost of evaluating the GP model, enabling faster decision making. Here, we describe advanced techniques that can be used to improve model accuracy with smaller datasets.

1. Kernel customization

a. Combining kernels. For more expressive behavior, multiple kernels can be combined into a single kernel through addition, multiplication, and tensor products. This combines the high-level functional behavior of expected phenomena into a single model. For example, when modeling beam size squared as a function of beamline parameters, the second-order dependence of beam size on quadrupole strength can be captured by a polynomial kernel, while a more general Matérn kernel can be used for other beamline parameters whose effect on beam size is less well known. However, this can come at a cost as more complex kernel functions can require more data to effectively learn the kernel hyperparameters, reducing sample efficiency if starting BO from scratch.

b. Hyperparameter priors. Kernel functions can also be customized by specifying prior distributions for kernel hyperparameters. Incorporating priors into the hyperparameter training process biases MLL hyperparameter training toward certain values according to the prior distribution. This provides a convenient middle ground between fixing hyperparameter values during optimization and training from scratch. For example, we can be relatively confident that linear beamline elements (such as quadrupole magnets) have a minimal effect on beam emittance. In this case, we can specify prior probabilities on the GP model length scales with respect to linear element parameters that encode this independence information into the GP model of the emittance, similar to what is done in the Sparse Axis Aligned Subspaces (SAAS) kernel [47]. Incorporating this assumption into the kernel rapidly speeds up convergence in objective functions that are strongly dependent on only a small subset of parameters by reducing the effective dimensionality of the problem. However, if nonlinearities

exist in these magnetic elements that do affect the beam emittance, the notion of independence in the GP model can be updated provided experimental evidence, instead of being ignored completely.

c. Kernel estimation from Hessian. Another way of encoding prior information of an objective function into the kernel can be specifying fixed kernel function hyperparameters that express expected functional correlations in a local region around the expected optimal point. For example, objective functions that depend on quadrupole strengths often contain a cross-correlation structure, similar to what is shown in Fig. 8(a), between adjacent quadrupoles due to the focusing-defocusing nature of first-order beam dynamics. These cross-correlations are difficult to learn on the fly without making a large number of measurements, as shown in Fig. 8(b). Adding information about cross-correlated structure in the objective function can significantly increase the accuracy of the GP model in high-dimensional spaces without having to make a large number of measurements. An efficient method for doing this is to compute the Hessian matrix of the objective function near the predicted optimal point in parameter space $\mathbf{x}^*$ [48]. This can then be used as the factor analysis distance metric described in Eq. (14), where $M = H_f(\mathbf{x}^*)$. As shown in Fig. 8(c), incorporating the Hessian matrix into the RBF kernel improves the accuracy of the GP model with fewer training data points. Identifying this low-dimensional structure in the objective leads to faster convergence speeds during optimization, especially in high-dimensional optimization problems [10].

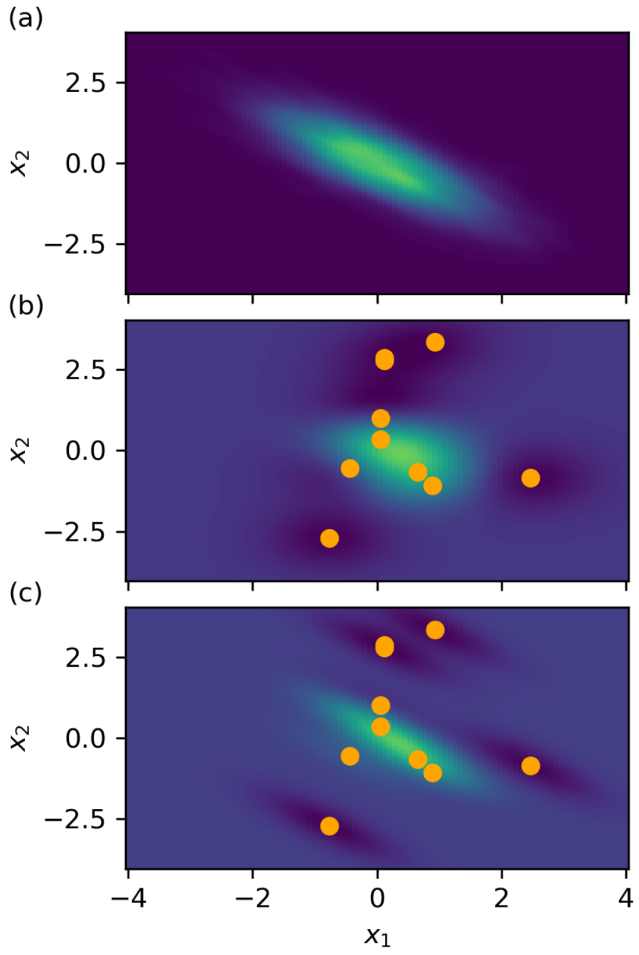

FIG. 8. Illustration of the improvement in prediction accuracy that can be gained by including expected correlations, such as those that arise from adjacent quadrupoles, into the GP kernel design. Here, a 2D function has input correlations that are similar to what one might observe between adjacent quadrupoles (a). For a fixed set of training data points (shown in orange), a GP model using an uncorrelated kernel (b) produces less accurate posterior predictions of the true function than a model with an accurate correlated kernel (c). In the context of BO, learning a more accurate model with fewer data training points translates to faster convergence in optimization.

d. Deep kernel learning. Neural networks (NN) can also be used as drop-in replacements for kernel functions in what is often referred to as deep kernel learning [49]. Neural networks can be incredibly powerful when modeling complex features in accelerator physics measurements such as images and signals. However, they require large or information-rich training datasets to accurately predict functional covariances between points in parameter space. As a result, learning kernel functions specified by NN on the fly during optimization is impractical for cases where measurements are expensive. Furthermore, for most optimization cases in accelerator physics, the objective functions are relatively smooth, thus much simpler kernel functions can reasonably predict accurate covariances without the cost of training an NN representation. As a result, there has been limited work in accelerator physics toward investigating the use of NN models in kernel functions for the purpose of conducting optimization. On the other hand, NN models have been investigated for use in other aspects of GP modeling, see Sec. III C 2 for details.

2. Nonconstant prior means

An alternative approach for incorporating prior information into the GP model is to specify an explicit, nonzero prior mean function. In this case, instead of embedding prior information about the high-level functional characteristics of the objective function, we provide an explicit guess as to what the objective function value is at every point in the parameter space. This is especially useful if we have detailed knowledge of the objective function from beam dynamics calculations, historical datasets, and/or previous optimization runs.

Incorporating a nonzero prior mean function $m(\mathbf{x})$ into a GP model reincorporates an extra term ignored in Eq. (9), producing posterior mean function values at the test points X_*

$$\boldsymbol{\mu}_* = \mathbf{m}(X_*) + K(X_*, X)K_y^{-1}[\mathbf{y} - \mathbf{m}(X)] \qquad (17)$$

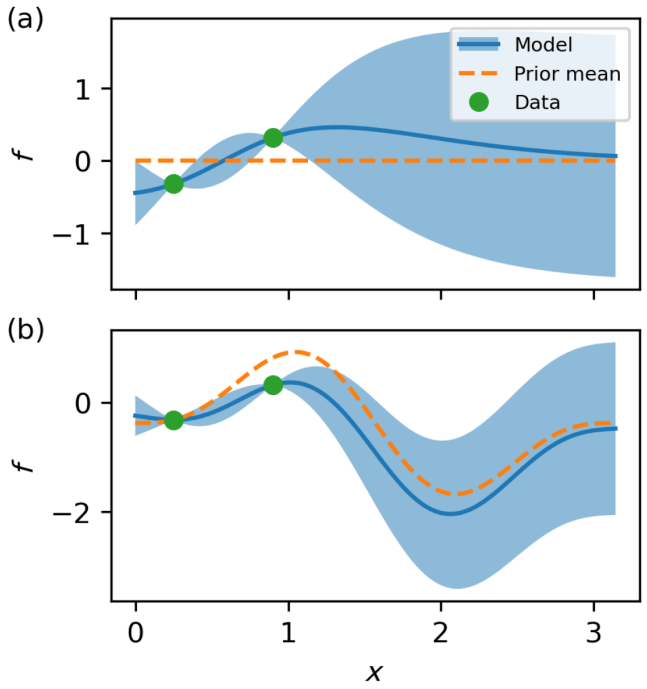

FIG. 9. Illustration of nonzero prior mean. In the absence of local data, the mean of the posterior distributions reverts to (a) zero or (b) the nonzero prior mean. The variance remains unchanged.

with $K_y = K(X,X) + \sigma_\epsilon^2 I$. For test points that are far away from previous measurements in parameter space ($K(X_*, X) \to 0$), the posterior mean function values $\mathbf{f}_*$ are equal to the prior mean values at the test points $\mathbf{m}(X_*)$. This effect is illustrated in Fig. 9, where the mean of the posterior distribution reverts back to the prior mean as the distance between test points and training data increases. Thus, if the prior mean function accurately predicts the objective function, the GP model can make similarly accurate predictions of the objective without any data. Conversely, if portions of the prior mean makes incorrect predictions, the posterior predictions of the GP model will reflect updated values from training data. In this way, the GP can be interpreted to model the difference to the prior mean function $m(\mathbf{x})$ instead of the full objective.

A custom mean function can be parameterized in a number of different ways, with the only requirement being that it maps parameter values to function values. For example, prior mean functions can be analytic functions, surrogate models, or even full particle dynamics simulations. However, for the best performance in the context of BO, prior mean functions should be differentiable (i.e., support backward automatic differentiation) and relatively inexpensive to evaluate. These attributes allow the acquisition function to be quickly optimized to find the next measurement point (see Sec. V for details).

For the purposes of online optimization in accelerators, neural network (NN) surrogate models [24,25,50,51] or fast differentiable beam dynamics simulations [52] have been identified as promising prior mean functions. Neural networks of sufficient complexity are known as universal function approximators [53] and typically execute much faster than conventional physics simulations [54]. Furthermore, NN models are generally differentiable due to training requirements, making them ideal for representing prior mean functions. Most importantly, unlike GP models that scale poorly with dataset size, NN surrogate model execution time is independent of the size of the dataset used to train the surrogate. This allows large amounts of historical measurement and simulation data to be incorporated into the GP model without hurting online performance. Alternatively, fast-executing beam dynamics simulations that are differentiable can be used to model the prior mean functions.

Incorporating prior mean functions inside GP models has a substantial impact on BO convergence speed. If the prior mean exactly matches the true objective function, convergence can happen immediately depending on the ratio of the objective function's ideal value with respect to the prior model uncertainty and which acquisition function is chosen to perform BO. Off-line studies using simulated objectives similar to those of the LCLS injector demonstrate that prior mean functions that have positive correlations with the true objective function also improve the convergence speed of BO to optimal values [50].

This benefit was also observed experimentally at the ATLAS accelerator [24] at Argonne National Laboratory. In this example, BO was used to tune the strengths of five quadrupole magnets to maximize the beam transmission through a beamline. They repeated the optimization multiple times starting with the same initial point using a constant prior mean function and three different NN surrogate models based on previous experimental data such that the models have varying correlations with the true objective function. Figure 10 shows that using a prior mean that has a high correlation with the ground truth resulted in better BO performance than standard GP models with constant priors. However, if the prior model is poorly correlated with the true objective function, then BO performance can suffer. Although measures can be taken to mitigate these effects [24], the quality of the prior mean is critical to improving the performance of BO when using a nonzero prior mean. Additional work at the LCLS photoinjector has demonstrated similar benefits to using NN surrogate models as priors in GP modeling up to nine free optimization parameters [24].

3. Modeling in transformed spaces

In some cases, it is advantageous to transform input or output data into an intermediate space before training hyperparameters and making model predictions. This strategy is useful when modeling objectives according to known physical principles or constraints. For example, a

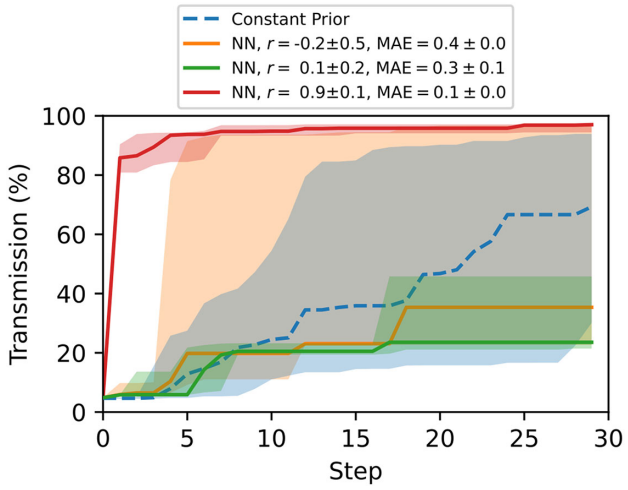

FIG. 10. Transmission optimization at ATLAS subsection using different prior mean functions. Solid and dashed lines depict the medians and the shaded areas depict the corresponding 90% confidence levels across 10 to 20 runs. Reproduced from [24].

number of objectives in accelerator physics are strictly positive, such as beam size and emittance. In order to restrict the range of GP predictions to positive values, data can be transformed into log space before fitting the GP model, as is shown in Fig. 11.

The *Bilog* transform [55] can improve modeling accuracy near zero by magnifying output values near zero and damping output values far away from zero. Applying this transform to data that is used to model constraint functions helps improve constrained optimization (see Sec. IV B 2), as constraints are generally defined with respect to zero, see Eq. (2). The *Gaussian copula* [56] has also been proposed as a useful transformation for magnifying objective function values that are on the edges of the observed range—namely maximum or minimum values.

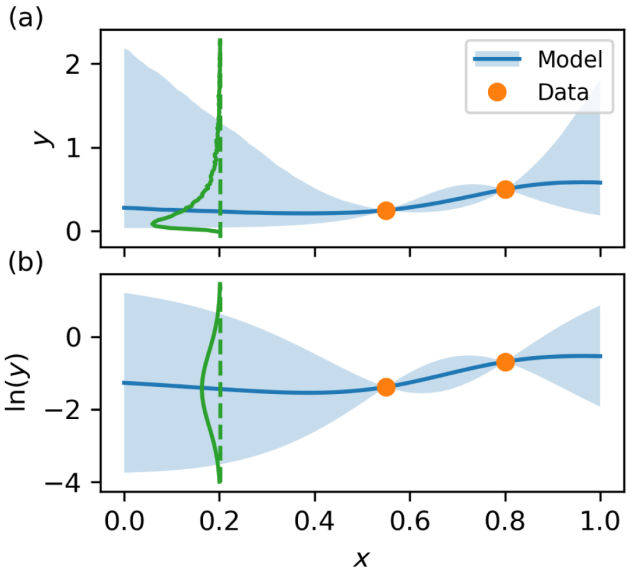

FIG. 11. Example of using log transformations in GP modeling for strictly positive output values. Data in real space (a) are transformed to log space before fitting a GP model (b). Samples drawn from the GP model in log space can then be transformed back into real space to make GP predictions. The resulting likelihood in real space is then a log-normal distribution which is strictly positive.

Depending on the modeling application, it may or may not be necessary to untransform GP model predictions in transformed space back into real space. For the logarithmic transformation example shown in Fig. 11, it is not necessary to untransform GP predictions back into real space, as optimizing the logarithm of the objective also optimizes the real objective value. However, in cases where untransforming model predictions back into real space is required by the application, these transformations can incur additional computational costs. Basic output transformations, such as data standardization or logarithmic transformers, allow for analytical calculations of the model mean and variance in untransformed spaces. However, more complex transformations may not have analytical inverses, precluding the use of analytical calculations of the model predictions in real space. As a result, model predictions in real space require Monte Carlo estimations of the posterior distribution, which incurs additional computational costs compared to analytical predictions. In this case, model visualization and acquisition function optimization may require more computational resources and time.

4. Time- and context-dependent modeling

In practical applications, the accelerator systems under consideration are often affected by factors beyond those represented in the input space, such as incoming beam parameters or drifts in auxiliary equipment due to external factors. While these factors cannot be controlled and changed explicitly by the optimization process, they can be incorporated into the GP model to improve the model predictive power and thus the convergence speed of BO. Using the kernel multiplication approach described in Sec. III C 1, the standard model can be extended with time and other contextual dimensions to represent a modified system

$$\mathbf{y} = f(\mathbf{x}, \boldsymbol{\phi}) + \varepsilon, \tag{18}$$

where $\boldsymbol{\phi}$ represents the contextual parameters. A classic use of such GP models has been in time-series predictions (i.e., stock prices), where $\boldsymbol{\phi}$ contains the time dimension t. This method is referred to as adaptive BO (ABO) [57] or contextual BO [58] and can be used to compensate for changes in the accelerator as long as they can be correlated to an observable. Note that to use such extended GP models in Bayesian optimization, all contextual parameters will need to be specified explicitly for the next point(s) and then held fixed during acquisition function optimization.

Incorporating additional dimensions into the GP model will increase the amount of data required for a good fit and thus slow down initial BO convergence (but not as much as a regular input parameter). ABO should only be used if the impact of contextual variables is significant relative to the noise in the objectives. Otherwise, it is advisable to use standard BO that will incorporate small drifts into the fitted noise parameter. For the most common case of time-adaptive BO, there are several choices of auxiliary variables that can be used—only time (which correlates to all drift sources but potentially has a complicated relationship that cannot be represented well by GP), time and specific drift sources, or only specific drift sources. Where possible, specific sources should be used to simplify the model. For example, if it is known that only room air and cooling water temperatures contribute to time-dependent drifts in rf cavities, it is best to only include temperature values as contextual variables instead of time. However, using time permits ABO to be very flexible with little to no tuning or when drift sources are distributed over too many dimensions.

Regardless of the choice of contextual variables, to achieve a good model fit with standard local kernels like RBF, the perturbations must be slow enough such that the BO loop can sample the highest-frequency features of the drift with at least a few points, by analogy to Nyquist's theorem. This requirement can be relaxed somewhat if a custom kernel is used that can account for long-range structure in the data. Experimentally, many drift signals are either directly correlated to something or have periodic structures. To avoid excessive hyperparameter tuning, special kernels like the spectral mixture kernel [59] can be used for cases where the exact number and periodicity of oscillatory signals are not known but require more data to achieve a good initial fit. A more advanced strategy is to use the rate of change of GP model parameters, such as length scales, with time to choose the most appropriate kernel so as to ensure an acceptable trade-off between worst-case performance and convergence. In Fig. 12, an example application to a linac trajectory stabilization problem is demonstrated. Performance of the more advanced methods strongly depends on drift magnitude, sampling rate, and measurement noise levels—it is suggested to perform similar simulations to evaluate suitability of contextual methods to specific tasks.

Both time and generic ABO has been demonstrated experimentally at the APS linac [26] and in the KARA storage ring [11]. Extensions of ABO to constrained problems with robustness requirements and dynamic kernel selection have also recently been tested at APS [27].

5. *Multifidelity modeling*

In some cases, it may be possible to obtain data about the behavior of the accelerator from different sources of information. For instance, we may have access to data from both experimental measurements and numerical simulations or from numerical simulations using different computational models and/or different resolutions. Additionally, we may be able to make experimental measurements that trade varying levels of detail and accuracy for evaluation speed or cost. In these cases, it is desirable for the GP model to be able to learn from these different sources of data while keeping track of their respective origin and evaluating their respective trustfulness.

In this context, the source of the data is encoded by assigning a *fidelity* value s to each data point in the dataset. Depending on the context, this fidelity parameter may take discrete values or continuous values. For instance, when combining experimental and simulation data (discrete fidelity), one may assign $s = 0$ to data points coming from simulations and $s = 1$ to data points from experimental measurements. When combining simulations at different resolutions (continuous fidelity), one may assign $s = 0$ to data points from low-resolution simulations, $s = 1$ to data points from high-resolution simulations, and an intermediate value of s to simulations at intermediate resolutions.

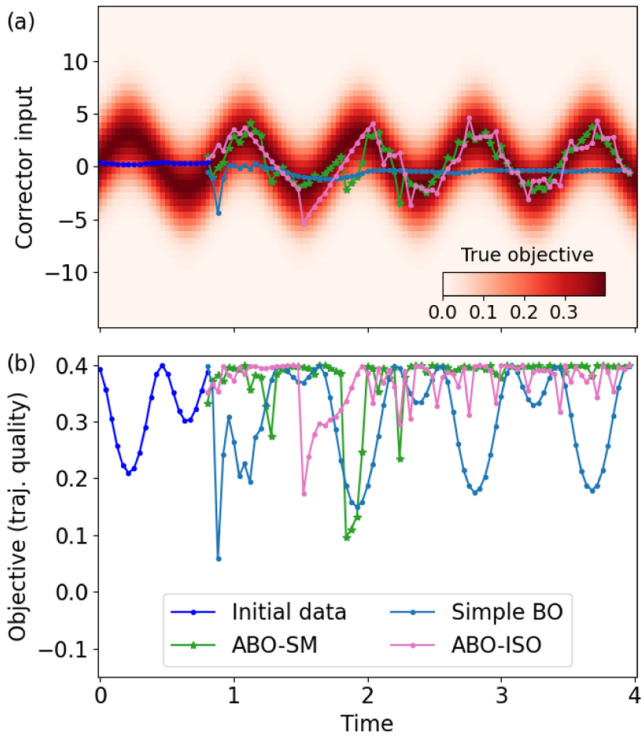

FIG. 12. Simulated application of standard and time-aware BO in a drifting trajectory stabilization problem. Simple BO settles on the mean value of the oscillations. ABO-ISO (isotropic) follows the changes but lags them because it only uses isotropic (local) kernel. ABO-SM (spectral mixture) captures long-range correlations and eventually correctly predicts necessary future changes in phase. By default, ABO-SM continues to explore around maximum value for optimization, producing a small step jitter. It can be eliminated by using the posterior mean as the acquisition function at the cost of convergence speed.

A GP can then be trained on this combined dataset, taking $\boldsymbol{x}$ *and* s as input and predicting the associated $\boldsymbol{y}$. In the case where s takes continuous values, the fidelity dimension is simply treated as another input dimension to the GP [60], with its associated kernel and hyperparameters. It is common to choose the kernel for s and $\boldsymbol{x}$ to be separable (see Sec. III C 1) and to use a stationary kernel for s [60]:

$$k[(s,\boldsymbol{x}),(s',\boldsymbol{x}')] = \tilde{\kappa}(||s-s'||)\kappa(\boldsymbol{x},\boldsymbol{x}').$$

In this case, the lengthscale hyperparameter l for the kernel $\tilde{\kappa}$ quantifies the extent to which similar fidelities give similar results. For instance, a large length scale l would cause the GP to predict similar output $\boldsymbol{y}$ even for relatively different values of the fidelity s. As usual, during training, this hyperparameter is often learned on the fly using hyperparameter tuning, and thus the GP automatically learns how much the prediction $\boldsymbol{y}$ varies with the fidelity s. Or, in other words, the GP learns to which extent the low-fidelity data can be relied on when trying to predict high-fidelity data.

In the case where the fidelity s is discrete, one type of multifidelity GP is the multitask GP [61], where the kernel is expressed in a similar manner:

$$k[(s,\boldsymbol{x}),(s',\boldsymbol{x}')] = \tilde{\kappa}_{s,s'}\kappa(\boldsymbol{x},\boldsymbol{x}'),$$

where $\tilde{\kappa}_{s,s'}$ is a positive semidefinite matrix (given that s and s' take discrete values). The values of the entries of this matrix are obtained by hyperparameter tuning, and they, again, quantify the extent to which low-fidelity and high-fidelity data are related.

This is illustrated with an example in Fig. 13, where low- and high-fidelity versions of an objective function can be evaluated with corresponding evaluation costs. Figure 13(a) shows a conventional GP model that predicts the output of the high-fidelity objective trained solely on a small, high-fidelity dataset. Instead of continuing to evaluate the expensive high-fidelity objective function, a multifidelity modeling approach incorporates inexpensive, low-fidelity data into the model of the high-fidelity objective. If the low-fidelity data serve as a good approximation of (i.e., is highly correlated with) the high-fidelity objective function, adding these data will reduce the uncertainty of the multifidelity model and increase its accuracy, as shown in Fig. 13(b). However, if the low-fidelity data are largely uncorrelated with the high-fidelity data, as in Fig. 13(c), the model prediction of the high-fidelity objective function is weakly influenced by the low-fidelity data.

Multifidelity GPs have been used in the context of simulation-based design optimization for laser-plasma accelerators [35,38]. In these instances, the fidelity was either continuous and corresponded to the resolution of the simulation grid [35] or discrete and corresponded to different simulation codes making different approximations [38].

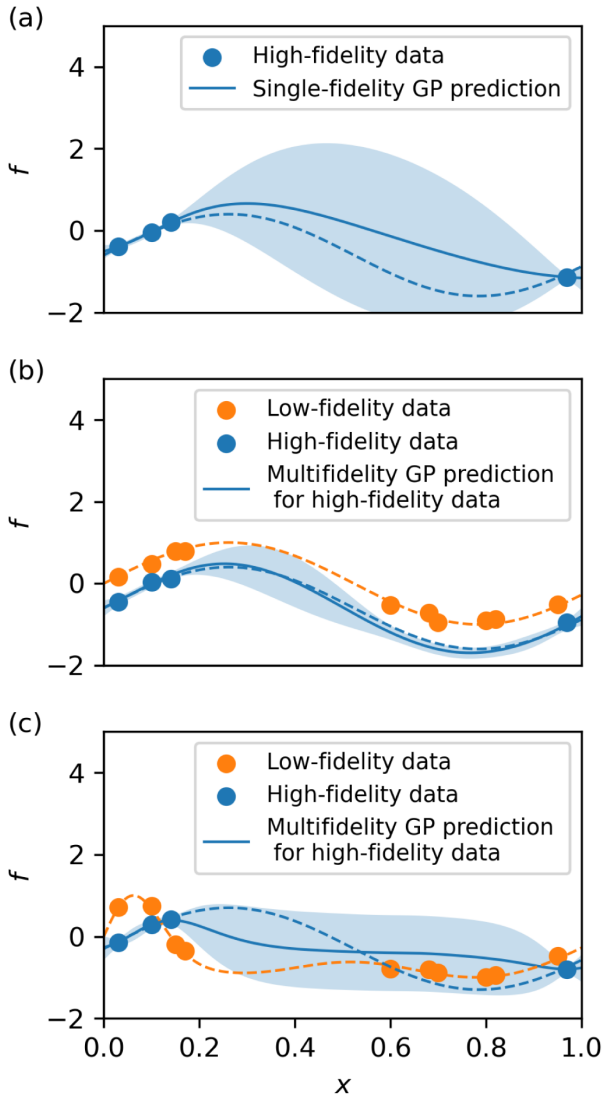

FIG. 13. Illustration of the prediction of a multifidelity Gaussian process, by comparing (a) a single-fidelity Gaussian process trained only on high-fidelity data, and (b),(c) a multifidelity Gaussian process trained on both high-fidelity and low-fidelity data, in the case where (b) high-fidelity and low-fidelity data are highly correlated, as well as (c) high-fidelity data and low-fidelity data are largely uncorrelated. In this particular example, the multifidelity GP is a multitask GP [61], as implemented in the library BoTorch. Dashed lines denote ground truth values of the low- and high-fidelity functions.

In both cases, the ability of multifidelity GP to partially rely on cheap, low-fidelity simulations (either low-resolution simulations, or approximated simulation codes) significantly reduced the cost of performing optimization. These examples are discussed in more detail in Sec. IV B 6.

6. Embedding complex modeling processes

As fast-executing surrogate models, GPs can be used as a drop-in replacement for other numerical models when

creating multicomponent models of complex systems. This can add flexibility, a robust treatment of uncertainty, and data-efficiency to arbitrary models of accelerator physics. These hybrid models, in turn, can increase the interpretability of GP modeling and in some cases expand the applicability of GP modeling to new domains.

For example, basic GP modeling is insufficient when describing systems that exhibit path-dependent physical processes, most notably, mechanical and magnetic hysteresis. Hysteresis is a path-dependent process such that beam properties depend not only on the current state of the machine but also on the historical path taken to get to the current state. This creates repeatability issues when optimizing accelerator parameters in magnetic and mechanical systems.

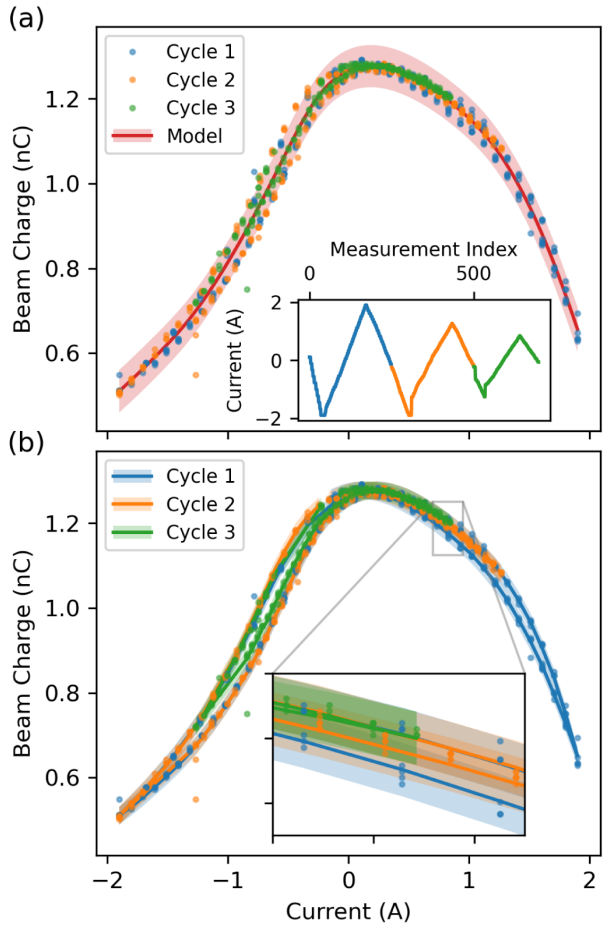

FIG. 14. Demonstration of combining GP models with a differentiable physics model of magnetic hysteresis. (a) Measured beam charge after passing through an aperture in the APS injector is plotted over three cycles of varying the current in an upstream quadrupole. Transmitted beam charge measurements are not repeatable due to hysteresis effects in the upstream quadrupole. (b) GP modeling with differentiable hysteresis model included accurately predicts beam charge over multiple hysteresis cycles with improved (reduced) uncertainty predictions. Reproduced from [28].

Basic GP modeling, see Fig. 14(a), interprets this error as stochastic noise, reducing the accuracy of model predictions, underestimating measurement uncertainty in some regions of parameter space, and overestimating uncertainty in others. However, if a GP model is combined with a numerical model of hysteresis, the hybrid model can make predictions with higher accuracy and better calibrated uncertainty estimates. In Fig. 14(b), a Preisach hysteresis model [62] is used to map the control parameter (magnet current) to magnetic field, while the GP model represents the mapping of magnetic field to beam dynamics [28]. Both hysteresis model parameters and the GP hyperparameters are trained simultaneously on the data using MLL. The resulting hybrid model has a higher predictive accuracy and provides uncertainty estimates that are well calibrated to stochastic experimental noise. As a result, the hybrid model improved optimization convergence, mitigating the detrimental effects of hysteresis on optimization when using basic GP models.

A key factor that enabled the simultaneous training of both hysteresis model parameters and GP hyperparameters was that calculations in both models were differentiable, allowing the use of gradient descent to maximize the MLL. Combining differentiable models of other nonrepeatable processes with GP models can extend the applicability of BO techniques to a wider range of optimization problems. Using GP models to represent smaller units of accelerator processes, as is done in the case here, increases their accuracy with smaller datasets and improves their interpretability.

IV. ACQUISITION FUNCTION DEFINITION

The acquisition function $\alpha(\mathbf{x})$ guides BO by defining the potential value of future measurements given a predictive surrogate model. During BO, input parameters that maximize the acquisition function will be chosen for evaluation during the next iteration.

Almost all acquisition functions aim to perform global optimization by balancing two optimization strategies, often referred to as "exploration" and "exploitation." Exploration places a high value on choosing points in parameter space that will add information to the GP surrogate model, often in regions of parameter space where the model has high uncertainty. Exploitation, on the other hand, places a high value on points in parameter space that the surrogate model predicts to be optimal. By balancing the weighting between these two strategies in the acquisition function (either implicitly or explicitly) during optimization, BO can increase the chances of efficiently finding global solutions to the optimization problem, instead of being stuck in local extrema.

As opposed to other standard optimization algorithm definitions, acquisition functions in BO are often defined with the assumption that objective functions are to be maximized. In order to use these acquisition functions for objective function minimization, transformations are

applied to the model predictions before they get passed to the acquisition function. This approach is preferable to modeling negated objective values with the GP, as it makes model interpretation more challenging.

In this section, we first describe basic acquisition functions used for general purpose single objective optimization. Then, we describe complex acquisition functions and modifications used to solve accelerator physics problems in online control and simulation.

A. Basic acquisition functions

The two most commonly used acquisition functions for performing optimization are expected improvement (EI) and upper confidence bound (UCB) [63]. These simple acquisition functions, illustrated in Fig. 15, are often the starting point for optimizing general problems and provide similar convergence speeds.

As its name suggests, EI uses the GP model to calculate an expectation value of the improvement $I(\mathbf{x}) = \max\{f(\mathbf{x}) - f(\mathbf{x}^*), 0\}$ over the best previously observed value of the objective function $f(\mathbf{x}^*)$. For a GP model with a Gaussian likelihood, the expected improvement can be calculated analytically:

$$\begin{aligned} EI(\mathbf{x}) &= \mathbb{E}[I(\mathbf{x})] \\ &= \sigma(\mathbf{x})[z\Phi(z) + \phi(z)] \\ z &= \frac{\mu(\mathbf{x}) - f(\mathbf{x}^*)}{\sigma(\mathbf{x})}, \end{aligned}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative density function and probability density function of a normal distribution. As shown in Fig. 15, EI emphasizes choosing observations that are predicted to be optimal, have a large variance, or a combination of both, thus balancing exploration and exploitation.

UCB explicitly specifies a trade-off between exploitation and exploration by using a linear combination of the predicted mean and variance from the GP model with a weighting factor β:

$$UCB(\mathbf{x}) \equiv \mu(\mathbf{x}) + \sqrt{\beta}\sigma(\mathbf{x}). \tag{19}$$

For general problems with an assumed smoothness, optimal values of β have been found to achieve the best convergence speeds [64]. Defining UCB with a larger β value favors exploration, while smaller values of β prioritize exploitation. If the objective function is expected to be convex or unimodal, smaller values of β may speed up convergence by prioritizing exploitation (often referred to as "greedy optimization").

EI and UCB often provide similar levels of convergence speed for most optimization problems. However, EI can sometimes become difficult to numerically optimize if large regions of the input space have zero probability of improving over the best observed point. In this case, gradient-based optimization of the acquisition function (see Sec. V) can struggle to escape regions with zero gradients, often referred to as the "vanishing gradient problem." However, it has been suggested that taking the log of the EI acquisition function before optimizing can address this issue [65].

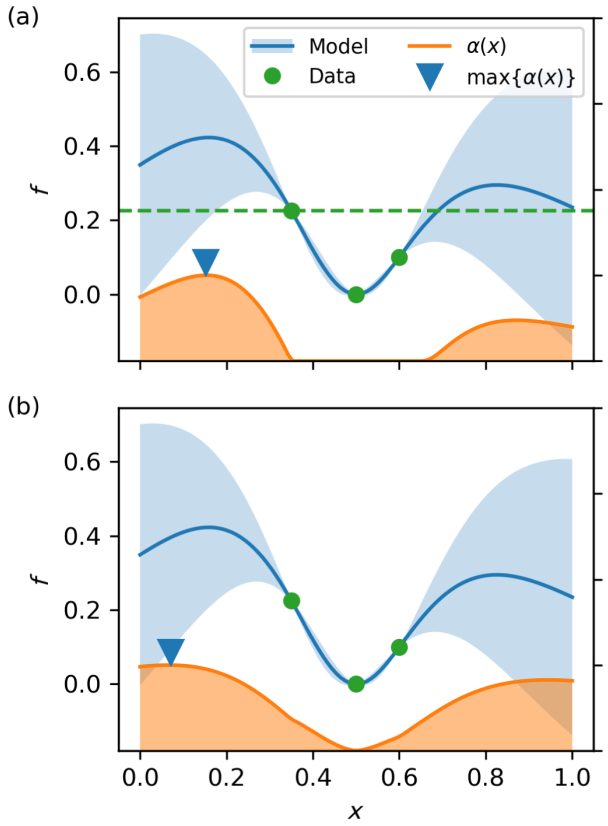

FIG. 15. Examples of the EI and UCB acquisition functions for objective function maximization given the same GP model and training data. (a) EI acquisition function, where the dashed horizontal line denotes the best previously observed value $f(\mathbf{x}^*)$. (b) UCB acquisition function.

B. Advanced acquisition functions

Here, we describe definitions and modifications that tailor the behavior of BO in order to solve problems in accelerator physics. In some cases, these acquisition functions are not analytically tractable and are thus evaluated by using Monte Carlo sampling. Calculations of the acquisition function in these cases are done by drawing function samples from the GP model and averaging over their individual contributions. A detailed discussion of this formalism can be found in [66].

1. Unknown function characterization

In some cases, instead of finding a solution to an optimization problem, we aim to characterize an unknown function to learn its structure and evaluate sensitivities to

individual parameters. So-called active learning acquisition functions can be defined to choose points that optimally characterize an unknown function instead of finding the extrema. A simple example of this is known as uncertainty sampling or "Bayesian exploration" (BE) [29]. In this case, the acquisition function is defined as

$$\alpha_{\mathrm{BE}}(x) = \sigma(x). \tag{20}$$

When using this acquisition function, BO will sample locations where the model uncertainty is maximized, usually at points in parameter space that are farthest from previous evaluation locations, as shown in Fig. 16(a). Combining this acquisition function with a GP model that uses automatic relevance determination (see Sec. III B 1) makes this technique especially powerful for performing high-dimensional characterization of previously unknown functions. In this case, the sampling pattern will change as the relative sensitivities of the target function with respect to each optimization parameter are learned, as shown in Fig. 16(b). Bayesian exploration and similar active-learning techniques have been used to perform a wide variety of characterization studies in both accelerator experiments and simulations [29], as well as in automating experiments in x-ray and neutron scattering experiments [67,68], and performing material discovery [69].

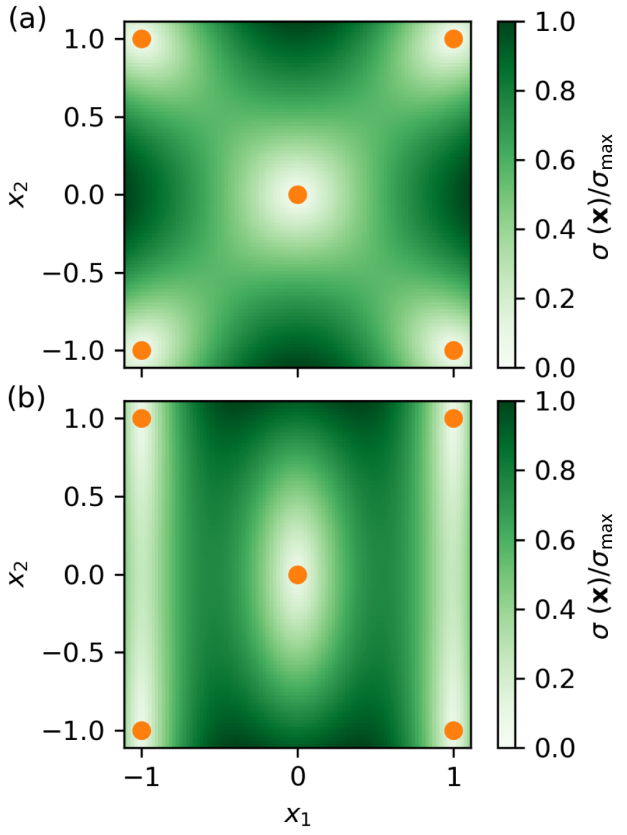

FIG. 16. Example of sampling behavior of Bayesian exploration (BE). (a) The BE acquisition function is maximized at locations in parameter space where the model uncertainty is highest, usually at locations farthest away from previous measurements. (b) In cases where the function is less sensitive to one parameter (x_2 in this example), the model uncertainty is smaller along that axis, resulting in less frequent sampling along that dimension.

2. Incorporating unknown constraints

Constraints play a crucial role in guaranteeing safe accelerator operation without damaging expensive equipment or increasing optimization time due to interlock violations. If the constraints are known, they can be incorporated by constraining optimization to a feasible subdomain of parameter space [70,71]. However, if the constraint functions $c_i(\mathbf{x})$ are unknown (as is often the case in accelerator physics), they need to be actively learned alongside the objective function $f(\mathbf{x})$ during optimization.

Several approaches have been developed in order to tackle the task of constrained optimization. These approaches aim to limit the number of times that constraints are violated during optimization, with varying degrees of "safety," referring to the likelihood that constraints are violated during optimization ("safer" algorithms are less likely to violate constraints). A review regarding safe optimization techniques, inside and outside the context of BO, is given in [72]. Two approaches of interest that have been used in accelerators are as follows.

The first approach is to modify the acquisition function by biasing it against selecting points that violate the set of constraints. This is done by weighting the acquisition function by the probability that the constraints are violated, calculated by integrating over GP models of the constraining functions [73]. While this method is straightforward to implement and interpret, it does come with some disadvantages. First, this formalism places a restriction on the unconstrained acquisition function, requiring that $\alpha(\mathbf{x}) \geq 0$, which is satisfied by most acquisition functions (with the notable exception of UCB). Second, in tightly constrained spaces, there are large regions of parameter space where the constrained acquisition function is nearly zero, thus resulting in large areas where the derivative of the acquisition function is also nearly zero, making it difficult to optimize with gradient descent methods. This issue can potentially be addressed by taking the log of the acquisition function prior to optimization as is suggested in [65]. Finally, while biasing the acquisition function in this way does minimize the probability that constraints are violated, it is possible that constraints will be violated during optimization. This technique has been applied to both online and off-line accelerator optimization problems with a variety of unconstrained acquisition functions, including single objective BO, Bayesian exploration [29,74] and multiobjective BO [34].

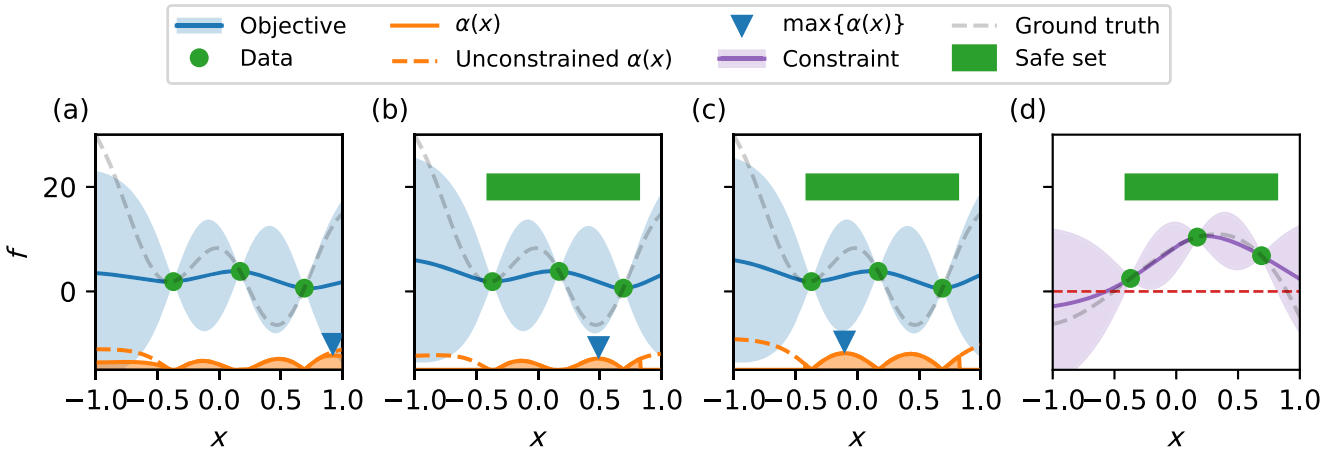

FIG. 17. Comparison between different constrained Bayesian optimization algorithms. (a) Weighting the acquisition function by the probability of satisfying the constraining function [73]. (b) Acquisition function optimization within a safe set using MoSaOpt in exploitation mode [32] and (c) SafeOpt [75]. (d) The constraint function, where valid regions satisfy $c(x) > 0$.

The second approach was originally presented under the name *SafeOpt* in [71,75]. Instead of modifying the acquisition function, predictions from the GP models of the constraints are used to define an arbitrarily shaped but compact safe set, within which the constraints are predicted to be satisfied with a desired confidence level under the assumption of Lipschitz continuity and knowledge of an upper bound of the Lipschitz constant [76]. Slightly less conservative but less intuitive conditions for the safety guarantees are given in the respective papers [71,75]. The acquisition function is then optimized inside this safe set, guaranteeing safety at a chosen confidence level. While in the original work, only a single constraint function was considered, multiple constraints can also be incorporated [71].

However, the safety guarantees provided by limiting the acquisition function optimization in this way come at a cost, as defining and optimizing over the irregular valid subdomain of parameter space are difficult, especially in high-dimensional spaces. Various methods have been introduced to improve the efficiency of defining the valid subdomain. The high dimensionality issue was addressed by *LineBO*, where the global BO problem is decomposed into a sequence of one-dimensional subproblems [30], reducing the memory required to represent the safe set at any one time. Stage-based procedures, i.e., *StageOpt* and *MoSaOpt*, where the expansion of the safe set (exploration) and exploitation phases, are staged [32,77]. This allows adjustment of hyperparameters in exploitation while still guaranteeing safety. Efficiency can be further improved by using goal-oriented safe exploration where expansion only takes place if necessary [78].

Given the safety guarantees, this branch of approaches originated in safety-critical fields such as robotics, but has also been successfully applied in the control of accelerators starting with [30]. Here, SafeOpt was applied for the beam intensity optimization at SwissFEL for up to 40 optimization variables. A lower threshold on pulse energy was considered for safety and the high dimensionality was addressed by using LineBO. Defining the valid sub-domain in a 1D space greatly simplified the problem and provided useful visual feedback to operators during optimization, without significantly degrading optimization performance.

Further adaptions to this application are made in [31], where in addition, application results to the High-Intensity Proton Accelerator (HIPA) are presented targeting to minimize the overall beam losses around the machine using 16 optimization variables and ensuring safety via 224 constraints coming from different interlocks. The stagewise procedure MoSaOpt was applied in simulation to the optical synchronization system as well as a laboratory setup at the European XFEL in [32] in order to minimize the timing jitter by tuning up to ten controller variables as optimization variables and ensuring an upper threshold on the timing jitter to maintain the stability of laser feedback systems.

A comparison of the algorithms used for performing constrained BO on a simple test problem is shown in Fig. 17. Figure 17(a) shows that weighting the acquisition reduces the chances of violating the constraint, although there are no guarantees that the constraint violations will not occur. On the other hand, methods that restrict the optimization of the acquisition function to within a valid subdomain of the parameter space, such as MoSaOpt [Fig. 17(b)] and SafeOpt [Fig. 17(c)], do not allow points that are predicted to violate the constraint to be sampled, ensuring safety.

It is important to note that both of these approaches to constrained optimization rely on accurate models of the constraining functions to effectively reduce the number of violations during optimization. As a result, most constraint violations happen during the initial stages of optimization,

where few observations of the constraining functions are available to create an accurate GP model. In order to prevent this, it is critical to start with a valid point in parameter space and conservatively explore the local region in the first initial steps or include prior information about the constraining functions into the GP model of the constraints.

Finally, it is reasonable to expect that concepts from the two methods currently used for constraining BO in accelerator physics can be combined into a single algorithm that contains the benefits provided by both methods. Additionally, the characterization of the trade-offs between safety tolerance and optimization speed should also be investigated.

3. *Multiobjective optimization*

In accelerator physics, it is often the goal of optimization to simultaneously minimize or maximize more than a single objective, referred to as multiobjective optimization. These objectives can compete with one another, requiring trade-offs between objectives to reach an optimal solution. For example, it is difficult to simultaneously maximize the dynamic aperture and the local momentum aperture of electron storage rings [79] or minimize the bunch size and beam emittance in a photoinjector due to space charge [6,54]. One strategy to solve this problem is to combine the objectives into a single objective by weighting the contribution of each objective to a single term, a process known as *scalarization*. However, the goal of multiobjective optimization is to determine the best trade-offs between multiple objectives, known as the *Pareto front* (PF). A PF represents a set of nondominated solutions, where no other solution can improve one objective without degrading at least one other objective. These solutions are considered *Pareto-optimal* because they form the best compromise among the multiple conflicting objectives.

One of the most popular methods for solving multiobjective optimization problems is the use of evolutionary algorithms [80], which use evolutionary heuristics to generate a large population of candidate points in parameter space from the previous generation to search for the PF. While these algorithms are easy to implement and use, they are incredibly inefficient, requiring the use of massively parallelized evaluation of many candidate points to converge to a solution set. As a result, multiobjective optimization is computationally expensive in the case of simulated optimization of beam dynamics and nearly impossible to use during beamline operations.

Multiobjective Bayesian optimization (MOBO) uses specialized acquisition functions to quickly identify the PF in multiobjective optimization problems. These acquisition functions rely on a metric known as the PF *hypervolume* (denoted $\mathcal{H}$), shown in Fig. 18(a). The hypervolume is a widely used quality indicator in multiobjective optimization and is particularly useful for problems with more than two

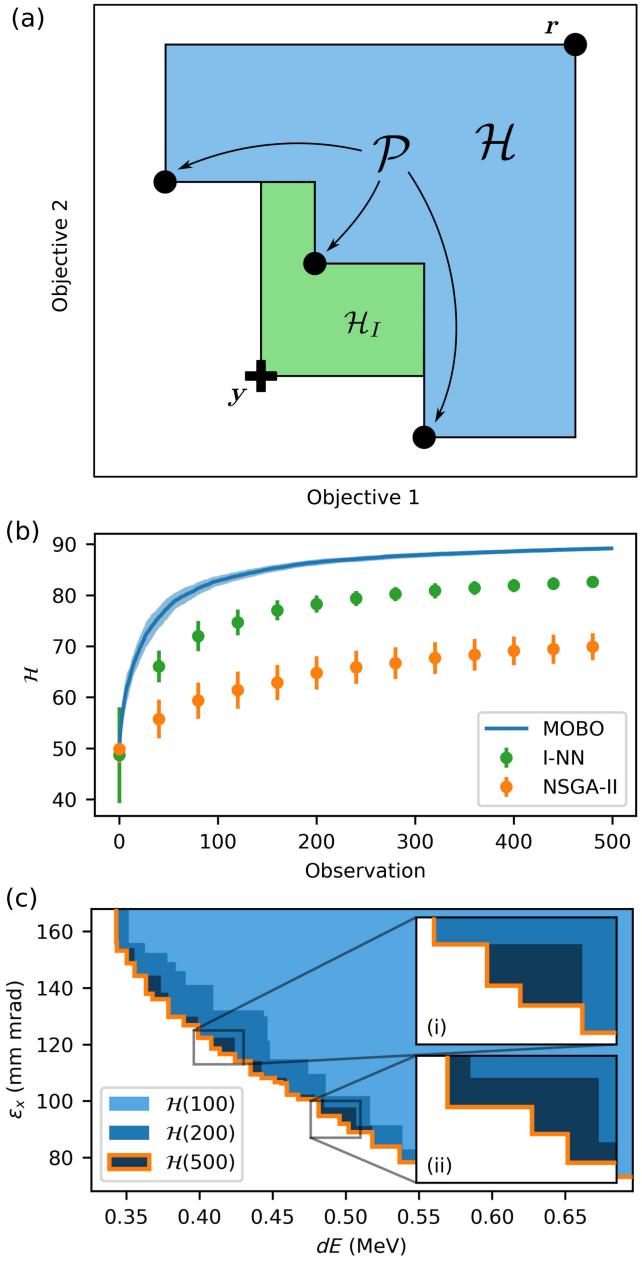

FIG. 18. Summary of multiobjective BO (MOBO) using expected hypervolume improvement (EHVI). (a) Given Pareto front $\mathcal{P}$ and corresponding hypervolume $\mathcal{H}$, the increase in hypervolume $\mathcal{H}_I$ due to a new measurement $\mathbf{y}$ is given by the shaded green area. (b) Comparison between multiobjective optimization algorithms for optimizing the AWA injector problem. NSGA-II is a standard evolutionary algorithm [81], I-NN is surrogate model assisted NSGA-II [54]. (c) Projected hypervolume after a set number of MOBO iterations with insets showing hypervolume improvement due to fill in points (i) and measurement of newly dominant points (ii). Reproduced from [34].

objectives. It measures the size of the dominated space, i.e., the portion of the objective space that is dominated by the PF. A larger hypervolume indicates a more accurate PF, as it dominates over a larger portion of objective function space

and thus a higher degree of Pareto optimality. To calculate the hypervolume, a reference point is specified in the objective space, typically set to be a point with the worst values for all objectives. Then, for each nondominated solution in the PF, the hypervolume is computed as the volume of the space dominated by the reference point and the current solution. The total hypervolume of the entire PF is the sum of these individual hypervolumes. Convergence of the hypervolume to a fixed value indicates that the true PF has been identified, as additional observations do not increase the quality of the PF.

The expected hypervolume improvement (EHVI) [82] acquisition function uses the notion of an increase in PF hypervolume to select points in parameter space. Starting with a PF containing previous measurements of the objectives, EHVI predicts the average expected increase in hypervolume [as shown in Fig. 18(a)] as a function of optimization parameters using GP models for each objective. As a result, BO using EHVI will select points that are more likely to maximally increase the hypervolume of the PF, whereas genetic algorithms select points only based on their Pareto optimality. When applied to identifying the PF of the AWA photoinjector containing seven objectives (three beam sizes, three beam emittances, and energy spread), EHVI was able to converge to a maximum hypervolume several orders of magnitude faster than evolutionary algorithms, as shown in Fig. 18(b).

EHVI is able to increase the PF hypervolume through two means, as shown in Fig. 18(c). In some cases EHVI "fills in" the multidimensional surface of the PF, leading to hypervolume increase that improves the detail described by the Pareto set. In other cases, EHVI increases the hypervolume by selecting observations that are predicted to dominate current nondominated points in the PF.

A major advantage of EHVI over genetic algorithms is that it can be used in serial optimization contexts where objectives cannot be evaluated in parallel, for example, determining the PF during online accelerator operations. However, a downside of the EHVI acquisition function is the computational expense associated with calculating the hypervolume improvement. The cost of partitioning the PF hypervolume into hyper-rectangles scales exponentially with the number of objective functions. As a result, computational costs can be a significant roadblock toward using EHVI in practice when a large number of objectives are to be optimized. This motivates the use of alternate acquisition function optimization algorithms in certain instances when a large number of objectives are present (see Sec. V for details).

A final consideration when using EHVI is the specification of the reference point. The reference point specifies the worst case value for each objective, thus any objective observations that are worse than the corresponding reference point values will not contribute to the PF hypervolume. As a result, the PF explored by EHVI will be limited to within the boundary specified by the reference point, thus ignoring objective function values beyond the reference point. However, specifying a reference point that is too far from the perceived optimal values of the objective functions will reduce the detail of the PF as different points will have vanishingly small contributions to the total hypervolume.

An example of using multiobjective Bayesian optimization experimentally was at the SLAC-MeV Ultrafast Electron Diffraction (UED) facility [33,83]. For MeV-UED, different scientific experiments often pose different requirements on multiple electron beam properties, such as electron pulse length, spot size at sample, and momentum space resolution (q resolution). However, it is difficult to simultaneously minimize these beam properties due to space charge forces. In practice, the evaluation time cost is high (∼60 s per data point) so that evolutionary algorithms are nearly impossible to use. The MOBO scheme is much more data efficient and can converge to the PF at least 1 order of magnitude faster than evolutionary algorithms. As a result, MOBO is the most suitable solution for beam optimizations at MeV-UED.

During optimization, gun phase and solenoid strengths were varied to explore the response of electron pulse length, spot size at sample, and q resolution and obtain PF giving trade-offs between them. MOBO was able to obtain PF within 150 measurements (∼3 h). The PF offers an unprecedented overview of the machine's performance limitations and can greatly assist human scientists in rapid decision making. The achieved performance was comparable with that obtained by experienced human operators and requires a significantly fewer measurements compared with traditional exploration methods such as a grid search (GS). During the experiment, the extra computation time associated with GP fitting and EHVI acquisition function optimization is small (below 5 s per iteration) relative to the reduction in beam property evaluation time associated with faster convergence of HV. Currently, the major limitation is the time taking electron beam diagnostics, by implementing highly efficient single-shot, nondestructive, and automated electron beam diagnostics, $>10^3$ data points could be obtained within a shorter time. This enhancement could improve the accuracy of GP and fully exploit the advantages of the MOBO algorithm.

4. Multipoint optimization and virtual objectives

In some optimization tasks, each acquisition requires a secondary scan in a separate domain to calculate the objective function. In engineering, this type of measurement process is referred to as a *multipoint query* (see, e.g., [84]). Consider, for example, the task of aligning particle beams through the magnetic center of quadrupole-focusing magnets. If the beam is misaligned with respect to the magnetic center of quadrupoles in the beamline, scanning the quadrupole strength results in a centroid kick causing

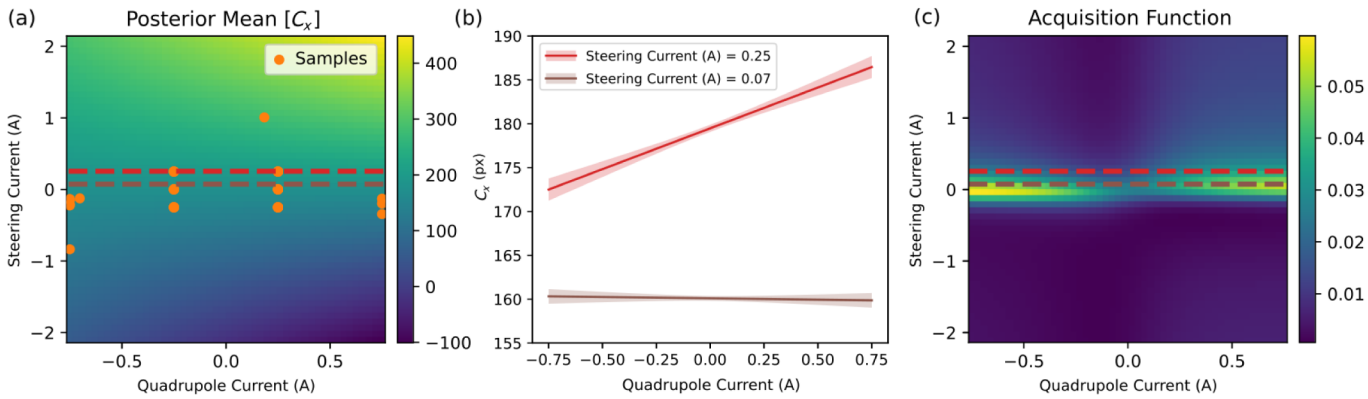

FIG. 19. Visualization of the BAX process for beam steering through quadrupole magnets. (a) Experimental measurements are used to build a GP model of the horizontal beam centroid position at a downstream screen C_x as a function of the quadrupole strength and steering parameter. Note that the GP model is built with a first-order polynomial kernel, constraining predictions to planar surfaces. Dashed lines denote cross sections of the GP model shown in (b). (c) The BAX acquisition function that predicts the information gained about the ideal steering current by making future measurements.

further misalignments downstream. This can be corrected through the use of steering magnets that provide an angular kick to the beam such that it intercepts the center of the quadrupole, removing the transverse kick due to changes in the quadrupole strength. However, determining the optimal steering strength requires either beam position monitors at the quadrupole location or constant scanning of the quadrupole strength while varying the steering parameter to estimate the beam misalignment. This is relatively simple for a single quadrupole but becomes increasingly complex when using multiple steering elements to align through multiple quadrupoles.

To address this problem using BO techniques, an acquisition function known as Bayesian Algorithm Execution (BAX) [85] has been developed which uses a so-called virtual objective to make control decisions. In the quadrupole alignment problem, the virtual objective to be minimized is the slope of the beam centroid with respect to the quadrupole strength, which is proportional to the beam misalignment. Instead of directly measuring this slope every time the steering parameter is varied, BAX builds a model of the beam centroid as a function of both the quadrupole strength and the steering parameter, as shown in Fig. 19(a). This model of the beam centroid is then used to predict the magnitude of centroid deflection as the quadrupole strength is varied (slope) as a function of the steering parameter, shown in Fig. 19(b). The BAX acquisition function uses these predictions to evaluate which future measurements will provide the most information about the steering current that leads to a minimization of the centroid deflection. This aspect is seen in Fig. 19(c), where the maximum of the acquisition function is at the edges of the quadrupole parameter domain (which provides the most information about the slope) and close to the optimal steering parameter. In the limit of many measurements, BAX will continue to make measurements close to the optimal steering parameter in order to improve model confidence in that region of parameter space.

This method of using virtual objectives can be extended to more complex situations. For example, performing alignment through multiple quadrupoles in both horizontal and vertical directions can be done by simply adding or multiplying multiple virtual objectives together into a single objective. BAX also supports more complex virtual objectives such as transverse beam emittance [37]. In this case, the virtual objective involves fitting polynomials to the beam size squared as a function of quadrupole strength using predictions from the GP model. At FACET-II, BAX was able to match the best emittance found by hand tuning, while at LCLS, the solution found by BAX produced about 25% lower emittance than hand tuning. In simulation studies, BAX minimizes the emittance using 20 times fewer beam size measurements than traditional BO. The dramatic improvement results from both increased sampling efficiency (by selecting single beam-size measurements at each acquisition) and modeling the beam-size function rather than the noisier emittance values.

5. Proximal biasing

Unlike optimization problems in other fields, online particle accelerator optimization sometimes requires incremental traversal of parameter space to maintain accelerator stability. Accelerator facilities often have many interconnected subsystems that are independently controlled through feedback systems to maintain accelerator parameters, such as water temperature, rf phase, and beam steering. As a result, making rapid changes in accelerator parameters can negatively affect these feedback loops, causing instabilities in accelerator operation that can ultimately shut down the accelerator. One strategy for mitigating this issue is to place a strict upper bound on

the travel distance from the current location in parameter space. Unfortunately, this in turn limits the exploration of parameter space needed to successfully find global extrema in BO. While it is possible (and sometimes necessary in sensitive systems) to place this hard limit on the maximum travel distance during each optimization step, it is sometimes more useful to bias the acquisition function toward making smaller steps in parameter space. This can be done through a technique known as "proximal biasing" [86]. Proximal biasing modifies a base acquisition function by adding a multiplicative term

$$\tilde{\alpha}(\mathbf{x}) = \alpha(\mathbf{x}) \exp\left(-\frac{(\mathbf{x}-\mathbf{x}_0)^2}{2l^2}\right), \tag{21}$$

where $\mathbf{x}_0$ was the last location in parameter space to be observed and l is an algorithm parameter that controls how strongly biased the acquisition function is toward making small steps in parameter space. This formalism places a restriction on the base acquisition function, requiring that $\alpha(\mathbf{x}) \geq 0$, however, is satisfied for most acquisition functions (again, with the notable exception of UCB).

A visualization of how proximal biasing affects BO is shown in Fig. 20. In this case, the goal is to characterize the first objective of the TNK test function [81] so the base acquisition function is Eq. (20). Figure 20(a) demonstrates that without proximal biasing, the exploration process makes large steps in parameter space in order to aggressively explore the objective function within the valid region. On the other hand, adding proximal biasing to the acquisition function significantly reduces the average step size, resulting in a smoother exploration of the parameter space, as shown in Fig. 20(b). In addition, proximal biasing does allow for larger steps in parameter space when necessary, as evidenced by the step highlighted by the green arrow in Fig. 20. If instead of proximal biasing, a hard limit on travel distance was set for this algorithm, it is likely that this larger travel distance would not have happened, resulting in a lack of exploration of the southernmost region of the valid domain.

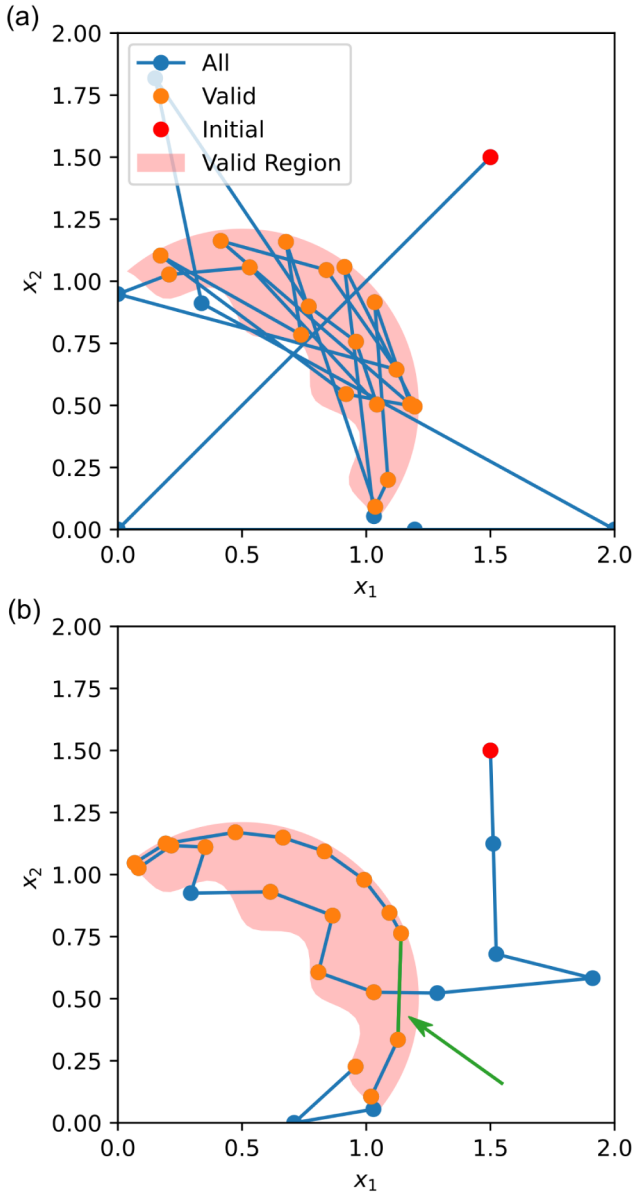

FIG. 20. Demonstration of proximal biasing effects during Bayesian exploration (BE) of the constrained TNK test problem. (a) Normal BE. (b) BE using proximal biasing with $l = 0.1$. The green arrow highlights a step where a larger jump in parameter space was allowed by proximal biasing. Reproduced from [86].

6. Multifidelity optimization

In the case where data can be queried at different fidelities (quantified by a parameter s; see Sec. III C 5), the BO algorithm needs to choose both the input parameter $\mathbf{x}$ and the fidelity s for each evaluation of the objective function. In addition to balancing exploration and exploitation, the algorithm must also balance the cost of an evaluation at a given fidelity with the corresponding information gain at the target (highest) fidelity. For instance, in the case where s represents the resolution of a numerical simulation, there is a trade-off between low-resolution simulations that provide low-fidelity information at a reduced computational cost and high-resolution simulations that provide high-fidelity information at an increased computational cost. If low-fidelity objective function values are strongly correlated with high-fidelity objective function values, BO can leverage low-fidelity approximations of the objective function to reduce the cost of optimization.

One simple way to handle this trade-off is to use repeated, fixed-size batches of low-fidelity and high-fidelity evaluations [87]. For example, multifidelity Bayesian optimization was run using a multitask GP model with repeated batches of 96 low-cost, low-fidelity simulations and 3 high-cost, high-fidelity simulations, in order to optimize the performance of a laser-plasma accelerator [38]. In this

case, the acquisition function was a modified version of EI [87], whereby only the highest-fidelity evaluations are considered when determining the optimal previously observed point $f(\boldsymbol{x}^*)$. In this particular example, single objective Bayesian optimization was observed to require 7× less computational resources to find an optimal accelerator configuration, compared to single-fidelity Bayesian optimization based only on high-cost, high-fidelity simulations [38].

However, in many cases, instead of using fixed-size batches of low-fidelity and high-fidelity evaluations, the fidelity s is dynamically decided by the algorithm for each evaluation. Typically, one would want the algorithm to mostly use low-fidelity evaluations early on in the optimization (to get a cheap, coarse picture of the overall objective landscape) and to progressively use more high-fidelity evaluations as it narrows down on the optimal point. It is also desirable that the algorithm rapidly stops using low-fidelity evaluations if the underlying multifidelity Gaussian process model determines that low-fidelity evaluations are not representative of high-fidelity data (see Sec. III C 5). This behavior can be obtained by using acquisition functions that incorporate both the cost and the information gain of an evaluation at a given fidelity; examples of such acquisition functions include multifidelity versions of upper confidence bound [60] and knowledge gradient [88]. An alternative to these modified acquisition function is to instead cast the multifidelity optimization as a multiobjective optimization problem [35,89]. In this scenario, a user-defined function assessing the reliability of a fidelity, denoted as s, is included as one of potentially multiple objectives. The expected hypervolume improvement (EHVI) acquisition function, explained in Sec. IV B 3, is used to solve this multiobjective problem, with an added penalty for the evaluation cost [89]. This multiobjective, multifidelity algorithm resulted in lower optimization costs when used in simulation-based design optimization of laser-plasma accelerators [35].

V. ACQUISITION FUNCTION OPTIMIZATION

Conducting BO involves addressing a numerical optimization challenge to determine the point in parameter space that maximizes the acquisition function. The computational demands of numerically optimizing the acquisition function make it the most resource-intensive step in the BO process. This process necessitates repetitive evaluations and/or sampling from the GP surrogate model posterior, incurring computational expenses–albeit generally less than those associated with evaluating the objective function directly. Adding to the complexity, acquisition functions are often nonconvex and may exhibit numerous local extrema [90]. As a result, the selection of the numerical optimization algorithm employed to optimize the acquisition function becomes pivotal for achieving optimal performance when using BO.

In scenarios where several points, or multiple objectives and constraints can be measured concurrently, BO can also be used to propose multiple measurement candidates. This is accomplished by specifying batched acquisition functions (referred to as "q sampling" [66]) that assign a joint utility to a set of q design points in parameter space. Alternatively, BO components can be combined with heuristics or other types of optimization algorithms (such as genetic algorithms) to find a large batch of potential points to measure simultaneously.

In this section, we highlight a variety of approaches to optimize acquisition functions, which affect the execution speed, improve performance of BO algorithms, and tailor BO to specific use cases.

A. Basic algorithms

The simplest approaches to optimizing acquisition functions are brute-force methods, such as random sampling or sampling on a mesh grid of points. These algorithms are usually poor choices for maximizing the acquisition function, due to their performance scaling to even modest numbers of free parameters. However, in low-dimensional parameter spaces (1–2 dimensions), the number of acquisition function evaluations necessary to maximize the acquisition function can be similar to other iterative methods due to their complex nature (nonconvexity). Given that the acquisition function can be evaluated in parallel through the use of batched computations, using random or grid-based sampling strategies can sometimes be faster than iterative optimization algorithms in this case.

Iterative, black-box optimization algorithms, such as Nelder-Mead simplex and RCDS can also be used to maximize the acquisition function. However, in most cases, maximizing the acquisition function is often best done using gradient-based optimization algorithms. The most straightforward example of this is using first-order gradient descent algorithms such as Adam [91]. Higher order gradient algorithms, such as L-BFGS-B [92], which uses an implicit estimation of the inverse Hessian, are also often commonly used to further speed up convergence.

In both cases, accurate calculations of the gradients can significantly reduce the number of iterations needed to reach convergence. Acquisition function calculations that are differentiable can be used to quickly calculate accurate gradients to speed up optimization. This is usually done by implementing the GP model and acquisition functions in a machine learning library that supports differentiability, such as PyTorch [93]. To improve the chances of finding the global maximum of the acquisition function using gradient descent methods, parallel optimization from multiple random starting points is often used to explore diverse regions of parameter space.

B. Trust region optimization

One disadvantage of BO is that common acquisition functions tend to overprioritize exploration over exploitation in high-dimensional parameter spaces. This is due to

the relatively large posterior uncertainties of GP models that result from the exponential growth of parameter space volume with dimensionality (models in high-dimensional space need more data to update prior function distributions). As a result, BO tends to pick points at the extremes of the domain in high-dimensional parameter spaces even if optimal points are found in a local region, see Fig. 20(a) for an example of this behavior. In addition, GP models used in BO aim to create a global description of the objective function, which may not be appropriate for functions that have varying local characteristics in different regions of parameter space.

Trust region BO (TuRBO) [94] aims to address both of these issues by restricting optimization of the acquisition function to within a so-called trust region around previous measurements where the model is expected to be the most accurate. The trust region is a local region centered at the best previously observed measurement so far during optimization, with side lengths equal to a base length L multiplied by the relative length scale of the GP model along each axis in parameter space. As optimization progresses, the location and size of the trust region are continuously updated to be centered at the best measured point in parameter space and scaled to match length scales of the GP model. Additionally, the base length of the trust region is increased or decreased based on the number of consecutive successes (improvements in the solution) or failures (no improvement), respectively. As a result, the trust region shrinks in cases where the model does not correctly identify the location of optimal solutions or expands the trust region when the model is making accurate predictions that result in continuous improvements in the objective function value. By limiting exploration of the parameter space within a local region, TuRBO transforms BO from a global optimization algorithm into a local one, resulting in substantially faster convergence to local extremum in high-dimensional parameter spaces than conventional BO.

A one-dimensional example of TuRBO applied to a test minimization problem is shown in Fig. 21. Despite large model uncertainties at the edge of the domain, which would normally cause BO to sample points on the boundary, TuRBO chooses observations that are in the local trust region around the best observed solution. In cases where the new observations do not improve over the best solution, the trust region contracts around the optimal point to increase model accuracy. If new observations do improve over the previous optimal point, the trust region is recentered at the location of those observations and expanded to find potential new solutions. Throughout the course of optimization, TuRBO will develop a locally accurate GP model near observed optimal solutions, instead of trying to accurately describe the global function behavior. While in this example TuRBO shrinks and expands the trust region after every step, a threshold for successes and failures is usually set such that multiple failures or successes in a row are necessary to change the overall trust region size.

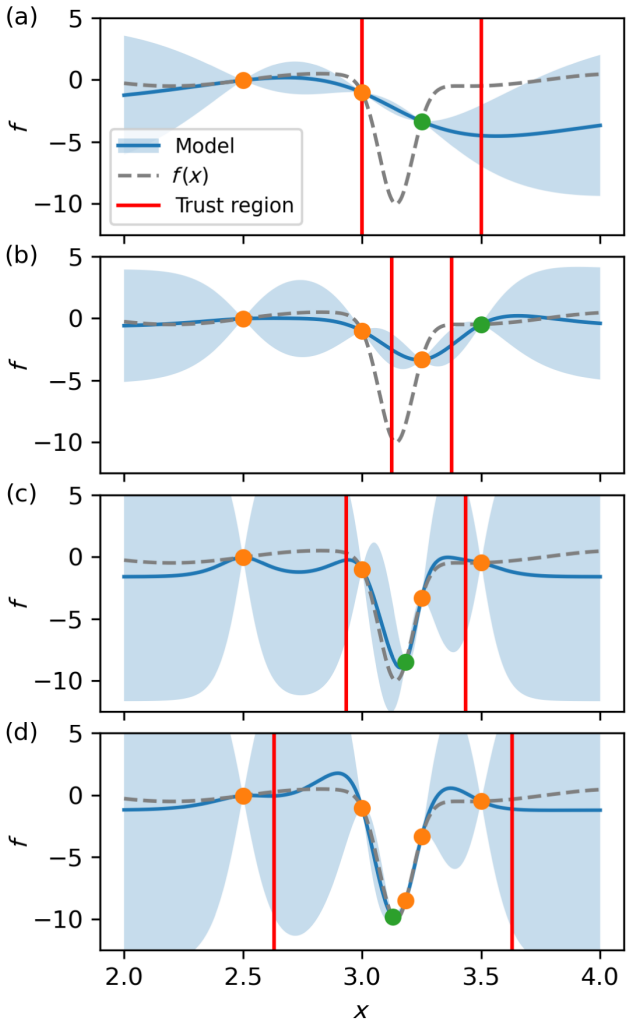

FIG. 21. One-dimensional visualization of trust region BO (TuRBO) applied to a minimization problem with the UCB acquisition function. (a)–(d) Sequential evolution of the GP model and sampling pattern. Orange circles denote objective function measurements and green circles denote the most recent sequential measurement at each step.

TuRBO was used on the ESRF-EBS storage ring [95] for the optimization of lifetime and compared to the existing optimization procedure. The 192 sextupoles and 64 octupoles available for the optimization of lifetime have been sorted and selected into 24 tuning parameters. To have fast and reproducible values for the optimization, the sum of all signals from the 128 beam loss detectors was used as the objective of the minimization rather than the lifetime value itself. Figure 22 shows the resulting lifetime during the optimization process performed with TuRBO, simplex, and UCB, all with similar initialization procedures. More details on the measurement can be found in [96]. TuRBO optimization was repeated 3 times and led in all cases to similar lifetime values within the same optimization time

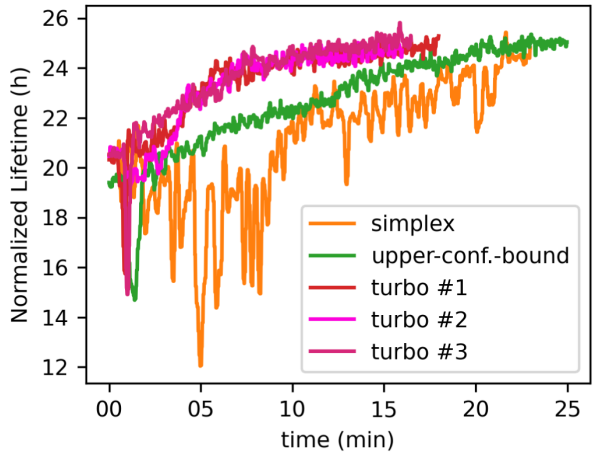

FIG. 22. Trust region BO (TuRBO), simplex, and UCB applied to the minimization of total losses (maximization of lifetime) at the ESRF-EBS storage ring. Adapted from [96].

and with comparable final sextupole and octupole settings. Additionally, it was found that TuRBO could quickly recover the optimal tuning configurations for the magnets from degraded storage ring conditions.

TuRBO can also be slightly modified to improve the exploration of tightly constrained problems where a majority of the input space violates one or more constraining functions. In this case, the goal is to reduce the number of constraint violations during optimization through the use of a conservative trust region. Instead of centering the trust region at the best observed solution, this approach centers the trust region at the barycenter of valid observation locations. Then the trust region side lengths are varied depending on the frequency of constraint violations observed during optimization or exploration. This prevents sampling at the extremes of the parameter space, which often results in measurements that violate the constraints.

C. Parallelized optimization

While in most cases BO is used in the context of serial optimization (one evaluation of the objectives/constraints is done at a time), it is possible for BO to propose a set of promising points that can be evaluated in parallel. We describe three distinct strategies here that are relevant to generating small ($n < 10$), medium ($10 < n < 100$), or large ($n > 100$) sets of candidate points to evaluate in parallel.

1. q sampling

This strategy aims to generate a set of candidate points in parameter space that jointly optimize given acquisition functions [97]. Many common acquisition functions (EI, UCB) can be expressed as the expectation of some real-valued function outputs at some designed input space [66]. Evaluating the acquisition function in the context of parallel selection of candidate points requires evaluating integrals over the posterior distributions. However, this makes evaluating parallelized versions of acquisition functions analytically intractable.

An alternative is to use Monte Carlo (MC) sampling to approximate the integrals. An MC approximation of acquisition function α at input space $\mathbf{x}$ using N MC samples given the data observed so far is

$$\alpha(\mathbf{x}) \approx \frac{1}{N}\sum_{i=1}^{N} a(\xi_i), \tag{22}$$

where $a(\cdot)$ is a real-valued function and the samples ξ_i are drawn from the posterior predictive distribution $p(y|\mathbf{x}, \mathcal{D})$.

To see this in action, we can examine the definition of parallelized expected improvement (qEI) [90,98,99] which generates q candidates that jointly optimize the EI acquisition function:

$$qEI(\mathbf{x}) \approx \frac{1}{N}\sum_{i=1}^{N} \max_{j=1,\ldots,q} \{\max(\xi_{ij} - f^*, 0)\}$$
$$\xi_i \sim p(f|\mathcal{D}), \tag{23}$$

where f^* is the best observed objective value so far.

To maintain the inexpensive computation of gradients for MC-based acquisition functions using automatic differentiation, a technique known as the "'reparameterization trick" [66] is used. Instead of sampling directly from the posterior of the GP model, samples are drawn from a unit normal distribution, then scaled and shifted such that the distribution matches GP predictions. This preserves differentiability by sidelining the stochastic generation of random samples.

2. Local penalization techniques

Local penalization is proposed as an alternative method for performing batched BO [100]. Instead of maximizing the joint distribution as in the q-sampling approach, it selects the samples in the batch sequentially and thus scales better with the input dimensions and batch sizes. The ith sample is selected by maximizing the product of the acquisition α and the penalization $\phi = \prod_1^{i-1} \phi_i$, where $\phi_i \in (0, 1]$ denotes the local penalization function around a previously selected point x_i in the batch. It effectively excludes the region around previously chosen points and goes to 1 elsewhere. The behavior of the penalization is governed by the Lipschitz constant of the objective function, which could be inferred from the GP model.

The local penalization method has been used in the simulation study at the linear accelerator FLUTE for radiation optimization [101]. It enabled an efficient selection of parameters to run parallelized simulations in a

high-performance computing cluster, resulting in better performance compared to using the genetic algorithm.

3. Large-scale parallelization

In cases where objective functions can be evaluated in a massively parallelized fashion (>100 simultaneous evaluations), i.e., in simulation on high-performance computing clusters, optimizing the acquisition function using the strategies outlined above may exceed the computational cost of evaluating the objective itself. As a result, it makes sense to use alternative methods for acquisition function optimization. Evolutionary or genetic algorithms are extensively employed toward solving optimization problems using large-scale parallelization. These algorithms use simple heuristics to generate candidate points, which is much cheaper than repeatedly numerically optimizing an acquisition function. Thus, it is advantageous to generate a large number of candidate points using a genetic algorithm and then determine a subset of those candidate points using a model-based acquisition function to be evaluated in BO. Combining genetic algorithms with BO takes advantage of both of their strengths, generating large sample sizes in a relatively short amount of time while still incorporating model information and acquisition function definitions into the selection of candidates for evaluation.

The multiobjective multigeneration Gaussian process optimizer (MG-GPO) represents one such algorithm that takes advantage of this combination [36,102]. This algorithm attempts to solve multiobjective optimization problems by first generating a number of candidate points using evolutionary heuristics (mutation [103] and crossover [104] operations). A subset of candidate points are then selected to be evaluated on the real objective by using a GP surrogate model (based on previous measurements or simulation results) to predict which candidate points are likely to dominate over previous measurements. By leveraging information in the learned GP surrogate model, the candidate points generated by MG-GPO are more likely to improve the Pareto optimal set when compared to model-free evolutionary algorithms (such as NSGA-II).

A slight modification can be made to MG-GPO to improve its performance by choosing a subset of candidates based on expected hypervolume improvement (as is done in conventional multiobjective BO) instead of predicted Pareto optimality. This has the added benefit of selecting candidates that not only will improve the Pareto front but also will maximize improvement according to the predicted increase in hypervolume once observed.

The MG-GPO method has been applied to design optimization of storage ring lattices [79]. It has also been applied experimentally to the SPEAR3 storage ring and the APS accelerator complex to demonstrate its online optimization capability with several important problems, including storage ring vertical emittance minimization with skew quadrupoles [102], nonlinear beam dynamics optimization with sextupoles [102,105], and linac front-end transmission tuning with steering and optics parameters [106]. In each case, it was shown that the algorithm can effectively improve the performance of the machine when compared to other algorithms.

VI. DISCUSSION

In this section, we discuss several aspects of BO that are relevant to its use in accelerator physics. We first describe the relationship between BO algorithms and other algorithms currently used in accelerator physics for optimization and control. We then discuss how to interpret and monitor BO performance during optimization and general best practices for improving optimization performance. Additionally, we highlight software packages, both inside and outside the accelerator physics field, that are used to implement BO algorithms. We also provide estimations of run time and computational memory usage for BO algorithms. Finally, we describe future research avenues in BO methods for accelerator physics.

A. BO in relation to other optimization algorithms

Here, we describe how classical BO relates to various other types of optimization and control algorithms. We also highlight the conceptual differences and similarities between online optimization and continuous control. Finally, we discuss the impact of different function approximations and ML model choices within those paradigms.

Note that we cannot make definitive, general statements about algorithm performance. The performance of a particular algorithm on a given accelerator problem is dependent on numerous factors, including, but not limited to, the specific algorithmic hyperparameters chosen, as well as the problem dimensionality, nonlinearity, convexity, multimodality, and noise.

1. Episodic optimization

Typically when describing "optimization," we mean an episodic process of adjusting settings to reach an optimal combination, which then ideally remains fixed for some period of time. Aside from BO, various other optimization algorithms have been developed and are actively used in the accelerator physics domain. Generally, these algorithms can be split into gradient-based and gradient-free (black box) algorithms, and, additionally, algorithms that learn some underlying representation of the system and those that do not.

Gradient-based algorithms use direct information about the gradient of the objective function or approximations of it (for example via finite difference methods) to determine setting changes during optimization. Gradient approximations on nondifferentiable systems (whether in simulation or in an experiment) can be time consuming to obtain, particularly as the number of variables increases. In some

instances in accelerators, gradient information has been approximated from machine jitter, allowing small, minimally invasive setting changes to slowly compensate for drift or move toward an optimum [107]. Gradient-based algorithms can easily become stuck in local minima, although techniques do exist to work around this (e.g., providing warm starts from a system model or previously known global solution, restarting the algorithm several times at different random starting points).

Gradient-based algorithms, such as stochastic gradient descent and variants (e.g., Adam, RMSProp [91,108,109]), can scale well to higher dimensions, particularly in cases where the evaluation of the objective function is fast and gradients are directly available. Consequently, they are used frequently in ML for training neural networks. In that context, updates to model parameters using small batches of data help to avoid local minima by adding noise to the gradient. Gradient-based methods can also be used in conjunction with differentiable models, e.g., through differentiable physics simulations [28,110–112], codes such as Bmad-X [113,114] or CHEETAH [52,115], or surrogate models based on function approximators such as neural networks [28,116].

Nelder-Mead simplex (NM) [117] is a gradient-free heuristic method that has been used extensively in accelerators for tuning [118–122]. It does not learn a model or use curve fitting but adjusts a "simplex" in search space at each iteration. NM requires very little preparation prior to use and is typically computationally inexpensive. For examples of studies that have run NM and BO on the same problem, see [10,12,48,123,124]. Theoretically speaking, NM is best suited to convex and noise-free objective functions [125], but it is difficult to assess how this translates to real-world experience in accelerators, where NM has performed well in practice even on quite noisy objectives.

Robust conjugate direction search (RCDS) [126] has been used for numerous accelerator tuning problems, particularly in rings for nonlinear dynamics optimization. In RCDS, local curve fitting at each iteration is used to aid estimation of the curvature of the objective function and the corresponding optimal direction in which to move settings. The addition of curve fitting adds robustness in the face of measurement noises and occasional machine failures. A successor variant RCDS-S [127] takes safety constraints and machine drifts into consideration.

A similar approach is taken in the BOBYQA algorithm, which constructs a second-order local model of function values near a candidate set of optimal parameters [128]. This algorithm has been used to perform optimization in simulation [129,130]. These approaches are similar to BO in the way that they create local models of the objective function to inform parameter selection for episodic optimization.

From the domain of feedback and control, extremum seeking (ES) has been applied to many accelerator problems [131–133]. ES adjusts settings with specific amplitudes and frequencies to approximate the gradient of the cost function and gradient descent, meaning that it works well as a local optimizer. Furthermore, ES parameter selection is much less expensive than BO methods, allowing it to be used to provide faster feedback than ABO approaches discussed in Sec. III C 4. However, ES can become stuck in local minima if not provided a sufficiently good starting point (e.g., provided by a system model [134]), and it does require careful adjustment of the main hyperparameters (the dither amplitude and frequency).

Finally, deep reinforcement learning (RL) has also found application in the accelerator domain [135–139]. While RL is traditionally used to train dynamic feedback controllers, it can also be used to train domain-specific optimization algorithms. In the case of RL, this may be referred to as reinforcement learning-trained optimization (RLO) [140]. Deep RL is computationally cheap and sample efficient at application time but requires significant upfront engineering effort to train. A case study comparing RL and BO on an accelerator tuning problem was conducted in [123].

2. Relation to continuous control and time-dependent control

By "continuous control," we refer to processes that are adjusting settings continuously as the accelerator is running (e.g., orbit feedback, corrections to low-level rf phases, and amplitudes to maintain the beam energy). A further distinction can be made between algorithms that take into account the sequential nature or time evolution of a problem and those that do not. In some classical control techniques such as model predictive control (MPC) [141] and in reinforcement learning (RL) [142], the sequentially dependent nature of a system is formalized as a Markov decision process [143], in which an observed system "state" is sufficient to predict the following system evolution. MPC and RL include direct consideration of the dynamic evolution of the system over a future time horizon when making decisions in the present. To accomplish this, these algorithms have access to or learn the dynamics of the system, and/or approximate solutions to the dynamic optimal control problem.

In contrast, classical BO assumes a stationary (i.e., nondrifting) system where the sequence of control actions is not taken into account in decision making. For example, when magnets are not affected by hysteresis, the problem of tuning magnets can be treated as nonsequential. When hysteresis effects are present, the sequence of magnet current settings affects the resultant magnetic field; as a result, the problem becomes sequential and this state information should be taken into account in decision making. Additionally, because BO is learning a stationary model of the objective function, its performance can degrade when being run on a nonstationary (i.e., drifting) system; this is why adjustments such as the adaptive BO approaches described in Sec. III C 4 are needed in order to run BO continuously as a feedback algorithm.

3. Relation to feed-forward corrections and warm starts

"Warm starts" or feed-forward corrections from learned models can be used both in continuous control and optimization in accelerators. For example, learned models can be used to provide fast setting changes when different setups are desired (e.g., see [116,134]), followed by fine-tuning with optimization algorithms such as BO. Indeed, the system model that provides the warm start can even be the GP model obtained from previous BO runs. Continuously running feed-forward corrections using ML models have also been used in accelerators; for example, this type of approach has been used for source size stabilization in light sources by compensating for optics deviations induced by different insertion devices [144].

B. Model choices

Bayesian optimization takes advantage of Gaussian process models, which can learn functions that are suitable for interpolation from very few samples and provide fairly robust uncertainty estimates, to perform efficient optimization of expensive objective functions. However, GP models do not scale as well as other model types, such as neural networks, to large datasets often required to solve high-dimensional optimization problems. As a result, they are more computationally expensive and typically slower to execute when solving high-dimensional optimization problems. For high-dimensional optimization and faster execution, BO can use other types of ML models so long as an uncertainty estimate is also available, including, but not limited to, Bayesian neural networks, quantile regression with neural networks, or neural network ensembles [49,108,145,146]. Using different types of surrogate models inside BO can also facilitate the inclusion of high-dimensional contextual information (such as initial beam images), which can improve convergence speed.

C. Interpreting BO performance

Unlike other optimization algorithms commonly used in accelerator physics, basic BO algorithms are designed to solve global optimization problems. This can sometimes lead to behaviors (shown in Fig. 23) that are unfamiliar to users expecting to see strong convergence to optimal values during optimization. Local optimization algorithms, such as Nelder-Mead simplex, often monotonically improve the objective function value, with small excursions around a local optimum to explore the objective function, as shown in Figs. 23(a) and 23(d). As we see in Fig. 23(a), this can sometimes lead to converging to a local optimum instead of the global one.

In contrast, BO algorithms often explore the domain to build a global model of the objective function in parameter space before sampling in a local region around the predicted optimal point. The number of iterations needed to perform this exploration can depend on the relative

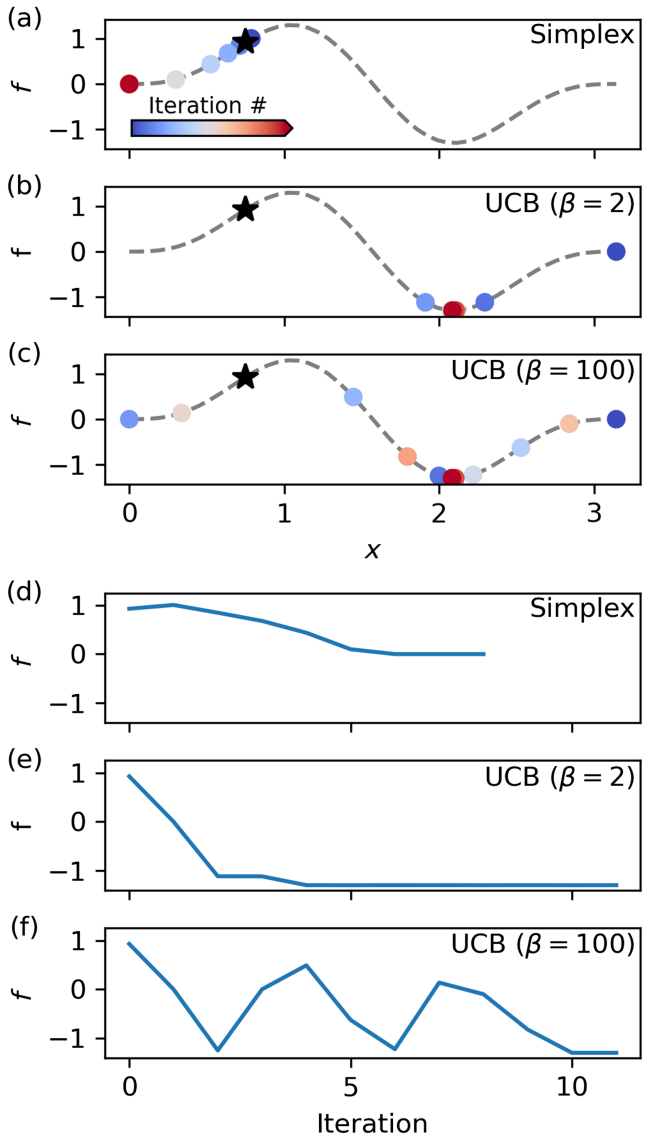

FIG. 23. Comparison of optimization performance between a local optimization algorithm (Nelder-Mead simplex), BO using the UCB acquisition function ($\beta = 2$), and BO using the UCB acquisition strongly weighted toward exploration ($\beta = 100$). All algorithms are initialized with a single observation at $x = 0.75$ and aim to minimize the objective function. (a)–(c) Observations of the objective function in parameter space for each algorithm. The dashed line denotes the true objective function. (d)–(f) Objective function values as a function of algorithm iteration. Note that simplex terminates after reaching a convergence criteria.

weighting of exploration vs exploitation in the acquisition function, the dimensionality of the parameter space, and the characteristics of the objective function. For example, when the UCB acquisition function is used with roughly even weighting between exploration and exploitation ($\beta = 2$), BO briefly explores parameter space before exploiting regions the GP model predicts are likely to be optimal, as shown in Figs. 23(b) and 23(e). Increasing weighting toward exploration, Figs. 23(c) and 23(f), increases the

number of iterations used to explore the objective function before it samples in the globally optimal region of parameter space. If the amplitude of the objective function far exceeds the predicted uncertainty by the GP model, exploration of the parameter space can cease relatively quickly compared to when optimizing more smoothly varying functions. Conversely, if the optimum of the objective function is comparable to the predicted uncertainty, strong convergence to the optimal value will occur once all other areas of parameter space have been explored. Increasing the dimensionality of input space further increases the number of iterations used to explore the objective function to build a global GP model.

As a result of the trade-off between exploration and exploitation, new users of BO algorithms who are used to seeing nearly monotonic improvements in the objective value might infer that BO optimization is performing poorly. However, it is important to keep in mind that this is a direct result of continuously searching for global extremum and does not signify an issue with the optimization algorithm. If the objective function is expected to be strongly convex, i.e., having a single global extremum, users can modify BO to strongly bias it toward exploitation through a variety of methods, including, for example, reducing the β parameter in UCB or utilizing trust region approaches such as TuRBO (see Sec. V B). In this case, strong convergence to a fixed location in parameter space is expected, at the cost of potentially converging to a local extremum.

D. Practical strategies for best performance

Here we discuss some best practices to improve the performance of BO methods in the field.

a. Normalizing training data. As is standard in most machine learning algorithms, it is critical that input data passed to the GP model is transformed prior to training in order to maintain the stability of hyperparameter optimization. It is standard practice to normalize the parameter space to the unit domain [0, 1] and to standardize objective function values such that they have a mean of zero and a unit standard deviation. Transforming training data in this way conditions the derivatives of the marginal log-likelihood with respect to hyperparameters to be of unit magnitudes, increasing the stability of gradient descent optimization of the hyperparameters.

Data that have been normalized and standardized are also more consistent with prior distributions inherent in GP models. The prior of a GP model is often stated as a distribution of functions with a zero mean and unit standard deviation. Having data that agree with this initial prior assumption also improves the robustness of maximizing the marginal log-likelihood as well as ensuring that covariance matrices are well conditioned. Additionally, it is often advantageous to place reasonable priors on hyperparameters such as the kernel length scale and likelihood noise to regularize hyperparameter training. Applying these priors to arbitrary modeling problems requires that incoming data are normalized and standardized.

b. Defining smoothly varying objectives and constraints. The accuracy of GP predictions relies on learned correlations between function values at different points in parameter space. As a result, BO works best when objective and constraining functions are relatively smooth or have some type of additional structure, such that these correlations exist. An example of where this becomes relevant in accelerator physics is maintaining a beam distribution inside a region of interest (ROI) on a diagnostic screen. One way to define this constraining function is to return a value of 1 if the beam is fully within the ROI and a zero otherwise. However, this is not ideal, as it is difficult for the GP model to predict where the boundary between valid and invalid measurements is given a limited set of data values (as function values in space are poorly correlated), as demonstrated in Fig. 24(a). On the other hand, if the constraining function measures how close the beam is to

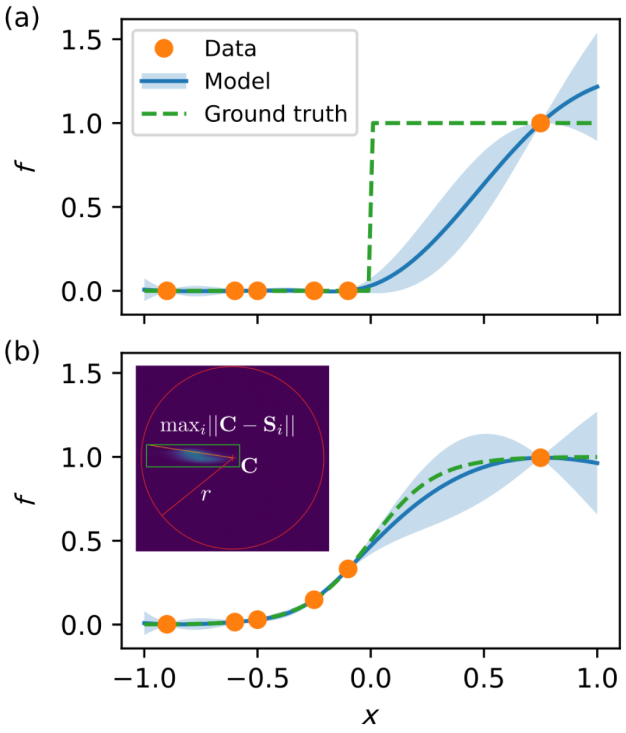

FIG. 24. Comparison between GP modeling of hard and soft constraining functions. (a) GP modeling of a heaviside constraining function does not accurately predict constraint values due to a single sharp feature that cannot be learned without dense sampling on either side of the constraint boundary. (b) Smooth constraining functions with a single characteristic length scale are more accurately modeled with GP modeling. Inset: Visualization of bounding box constraint function $f(x) = \max_i\{||\mathbf{C} - \mathbf{S}_i(x)||\}$ used to keep beam distributions inside an ROI, where r is the radius of a circular ROI, $\mathbf{C}$ is the center coordinates of the ROI, and $\mathbf{S}_i$ are corner coordinates of a bounding box around the beam.

violating the constraint, as shown in Fig. 24(b) and discussed in [74], the GP model can accurately predict extrapolated constraining function values with fewer measurements, which reduces the number of constraint violations during optimization.

c. Interpolated measurements. In some instances during accelerator operations, it is faster to make multiple measurements of the beam distribution than making control decisions using BO algorithms. Additionally, modifying the accelerator control parameters is more time consuming than making measurements, especially if large changes in parameter values take a longer amount of time. This is often the case when tuning magnet parameters, as power supplies take a nonzero amount of time to change the applied current, or in mechanical actuators, where stepper motors take time to traverse the operational range. Bayesian optimization can reduce the costs of making large changes in input space when performing optimization (see Sec. IV B 5), however, this requires reoptimizing the acquisition function at every step, which can become costly.

An alternative approach to minimizing measurement costs on the total optimization cost is to precompute a set of evaluation candidates along the path of tuning parameters. Instead of immediately jumping to the next point proposed by BO during optimization, we can generate a set of future evaluation points that interpolate between the current set point and the future set point, as shown in Fig. 25. This allows multiple measurements to be taken quickly throughout the input space without the need to wait long periods of time for accelerator parameters to change or to reoptimize the acquisition function for each measurement. As a result, the GP model contains additional training data after each optimization step, enabling BO to make better decisions with fewer optimization iterations. This does, however, increase the computation time associated with training the GP model and optimizing the acquisition function, requiring careful consideration of the trade-offs associated with using this technique.

d. Leveraging batch computations. To address modern challenges in high-performance computing, significant efforts in the machine learning community have been directed towards developing advanced hardware and software solutions for rapid matrix calculations. For example, graphics processing units (GPUs) are specifically designed to perform expensive matrix computations extremely quickly using massive hardware parallelizations. Bayesian optimization algorithms are well suited to take full advantage of these developments as most computations involved in making GP predictions involve matrix manipulations. Extending the evaluation of GP models or acquisition functions in parallel using *batched computations* (which adds new dimensions to matrices used in evaluations) plays a critical role in leveraging modern computing hardware and software to improve performance.

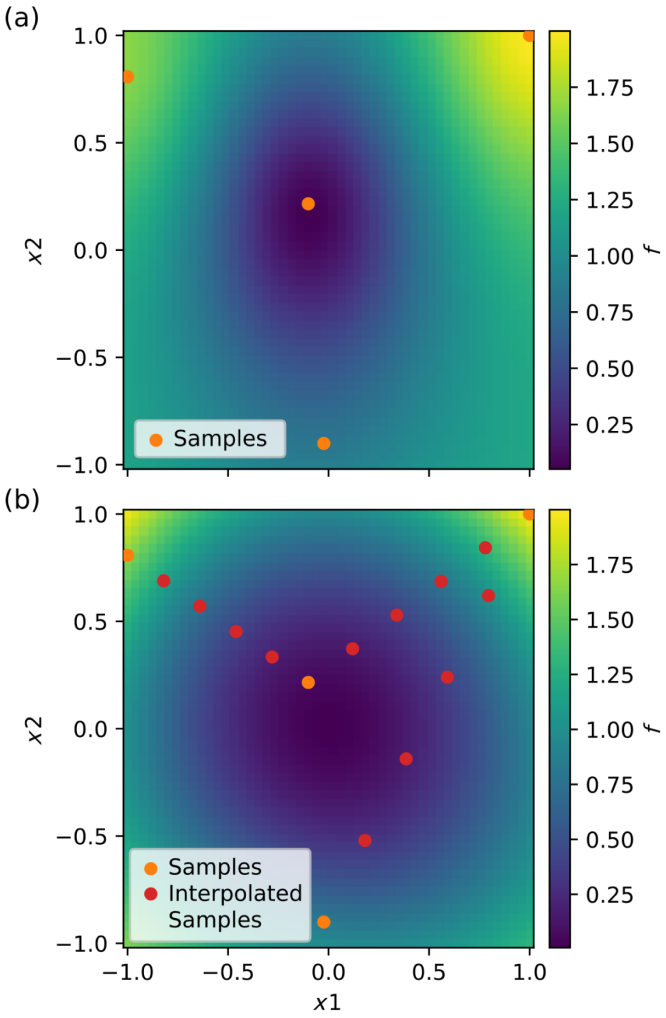

FIG. 25. Comparison between GP modeling of the two-dimensional sphere function $f(x_1, x_2) = x_1^2 + x_2^2$ with and without interpolated measurements. (a) Shows the posterior mean of the GP model with four measurements taken sequentially. (b) Shows the same four measurements taken sequentially but with interpolated points in between each measurement. Incorporating interpolated points in the dataset leads to higher modeling accuracy, leading to accurate identification of the sphere function minimum at the origin.

A core application of batched computation is acquisition function optimization. Optimizing acquisition functions is often a challenging problem, as they usually are not convex and can contain many local extrema. Multirestart optimization can be used in this case to improve the search for a global maximum by restarting optimization at a number of different initial starting points. Batched computation allows this process to happen in parallel, significantly reducing the computation time needed to maximize the acquisition function while leveraging the advantages provided by fast matrix computational techniques. As a result, high-performance software libraries that implement BO take advantage of this technique (see Sec. VI E).

E. Implementations of BO

There are several open-source software packages that implement GP modeling and BO, mostly using the Python programming language for as a programming interface and C/C++ for computations. It is strongly recommended that practitioners of BO do not "reinvent the wheel" when trying to implement BO algorithms to solve their specific optimization problems. Current implementations of BO (as described below) have capabilities that cover a wide range of accelerator physics problems and applications. If further modifications are needed to tackle a specific problem, it is strongly recommended that these modifications are built from existing software packages with the intent to contribute back to the existing package for others in the community to use. This will accelerate the state-of-the-art for all parties and prevent a fractured landscape of competing implementations that hinder algorithmic development and application to optimization problems.

a. Scikit-learn. The Scikit-learn general machine learning package [147] provides a simple implementation of basic GP modeling and BO while also providing good documentation, making it a good resource for gaining experience using basic BO algorithms.

b. BoTorch, GPyTorch, and Ax. This set of open-source packages is developed and maintained by Cornell University, Columbia University, University of Pennsylvania, New York University, and Meta [148] and provide implementations of state-of-the-art GP modeling and BO. They are built upon the PyTorch [93] machine learning language that implements automatic differentiation and GPU computing, both of which significantly improve the speed and performance of BO. BoTorch relies on a lower level package, GPyTorch [149], to implement GP models, allowing significant customization of all aspects of GP modeling, including custom kernels, priors, and likelihoods. BoTorch also takes advantage of batched Monte Carlo sampling to maximize performance when computing and optimizing acquisition functions. With the recent improvement in PyTorch like the JIT compiler, BoTorch stack is very competitive in performance benchmarks and is highly amenable to GPU acceleration. BoTorch is complemented by Ax, which provides an accessible user interface to BoTorch.

c. Xopt The Xopt Python package [150,151] is a high-level optimization package that connects advanced optimization algorithms to arbitrary optimization tasks (in both simulations and experiments), with a focus on solving problems in accelerator physics. The object-oriented structure of Xopt allows for significant flexibility in defining and executing optimization processes, including specification of optimization runs through simple text files (YAML, JSON), asynchronous evaluation of objective functions, model introspection during optimization, and human-in-the-loop optimization. Xopt implements and adapts existing implementations of several algorithms for easy off-the-shelf use in accelerators, including most of the BO algorithms and techniques discussed in this review. These algorithms can be easily tailored toward solving specific optimization problems through the use of subclassing. Xopt has been developed by the SLAC machine learning (ML) group specifically to address optimization problems in accelerator science, with the ability for extension and customization for other scientific fields. It has been used to perform online accelerator control at a number of facilities including LCLS, LCLS-II, FACET-II (SLAC), AWA, ATLAS (Argonne), FLASH, FLASHForward, European XFEL, Petra-III (DESY), RHIC, NSLS-II (BNL), ESRF, and LBNL. It has also been used to perform optimization in simulation at Cornell University, University of Chicago, and on high-performance computing (HPC) clusters such as NERSC.

d. Badger. The Badger package [119,152], also developed by the SLAC ML group as a successor to DESY's *Ocelot Optimizer* [120], provides an easy-to-use graphical user interface for accelerator control rooms to interface with algorithms implemented by Xopt. It provides an extendable interface for communicating with a variety of accelerator control systems and can be customized with extensions to provide online analysis of optimization performance and algorithm introspection.

e. Optimas. The Optimas package [38], focuses on optimization workflows using numerical simulations at varying computational scales, from laptops to high-performance computing platforms. Optimas relies on the library libEnsemble [153] to orchestrate multiple simulations running concurrently as part of the optimization, and to allocate appropriate multi-CPU and multi-GPU resources to each of these simulations (as well as GPU resources, if needed, for the Bayesian optimizer). Optimas provides multiple algorithms for parallel parameter exploration and optimization such as single- and multiobjective Bayesian optimization, including multifidelity and multitask options. It is also highly interoperable with the Ax library. Optimas has been used on large-scale clusters such as Perlmutter (NERSC) and JUWELS (JSC) and is developed by a collaboration between DESY, Lawrence Berkeley National Laboratory, and Argonne National Laboratory.

f. APSopt. The APSopt package [154] is being developed by the APS accelerator physics and operations group to integrate internally and externally developed BO, RL, and classical methods into a robustly tested tool for both API-based use by physics experts and GUI-only use by operators. It aims to provide a coherent optimization environment through a number of advanced features for data management, distributed client-server operation, automatic initialization with machine-tuned algorithms, parameter hints, and human-in-the-loop interactive model review and refinement. It has been experimentally tested in the APS injector, APS storage ring, NSLS-II storage ring, Fermilab IOTA/FAST complex and is being used extensively for the APS-Upgrade commissioning.

g. GeOFF. The generic optimization framework (GeOFF) [155] is being developed by the data science

teams at CERN and GSI. It allows to easily integrate RL, BO, and numerical optimization in the control room or for off-line optimization on, e.g., simulation. The optimization problem definition can handle arbitrarily complex controls or simulation processes as long as a Python interface is available. GeOFF also comes with a plug-and-play graphical user interface to test and run the optimization or continuous control problems. It is routinely used at CERN for the entire accelerator complex and for various problems at GSI.

F. Computational requirements and scaling

Because of the complexity involved in efficiently implementing the low-level mathematical routines, the above BO packages rely on linear algebra or machine learning frameworks, predominantly PyTorch. While the theoretical complexity and storage scaling of BO methods is well understood and has been discussed previously, in practice, the performance of the PyTorch/GPyTorch/BoTorch libraries can deviate significantly from theoretical behavior.

Due to the high costs of the specialized GPU hardware, it is critical to understand what tasks are computationally feasible in practice on CPUs and GPUs. In Fig. 26, we provide benchmark results for a "midrange" 2023 ML hardware setup consisting of 16 cores (32 threads) from AMD EPYC 7742 CPU and a single A100–40 GB GPU. We do not consider multi-GPU configurations, but they are supported by PyTorch—this yields only a slight increase in the feasible problem sizes.

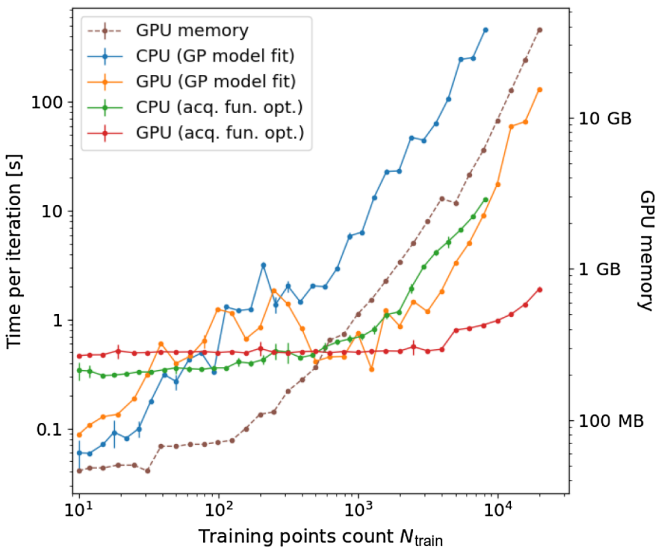

FIG. 26. Performance scaling with dataset size for BoTorch/GPyTorch (0.9.4/1.11) libraries on a single-objective optimization run. Synthetic five-variable quadratic objective was used with Monte Carlo version of UCB acquisition function and 100 Adam optimizer iterations. GPU memory usage is only applicable to GPU runs.

We benchmark three typical components of the BO process—GP model fitting, GP model evaluation, and acquisition function optimization (which involves model evaluation and auto-grad operations as part of the optimizer loop). The overall scaling is consistent with expectations, with model fitting and evaluation showing $\mathcal{O}(n^3)$ growth as a function of number of collected points n. However, the progression is not smooth, with repeatable deviations at particular sizes due to different bottlenecks and code paths that are encountered depending on internal PyTorch configuration. Note also that there is a constant time floor of 100–1000 ms per BO loop due to initialization, data copies, and Python overhead—in practice, this limits BO applications to make sub-1 Hz decisions (although data acquisition can take place at a higher rate) Using more complex modeling architectures (such as incorporating nonconstant prior means) can further reduce the speed at which decisions are made by the BO algorithm. Implementing BO on specialized computing hardware such as application-specific integrated circuits (ASICs), field programmable gate arrays (FPGAs) or HPCs could speed up BO in these cases.

The ultimate limit on the number of model points is determined by available memory and is encountered at ~25 k points on a 40 GB GPU (at which point the CPU is too slow even if there is sufficient RAM). Approximate GP methods can extend this limit but are not particularly popular in BO applications. Our practical recommendation is to limit problem sizes to 10k points with a GPU and 3k with a CPU-only machine and apply BO only in cases when objective evaluation time is sufficiently long to amortize computational costs for your particular choice of model, acquisition function, and hyperparameters (see Sec. II B). This ensures that BO use is worthwhile in terms of overall wall-clock convergence speed.

G. Future directions for BO research in accelerator science

While BO algorithms have been shown to be able to solve a wide variety of accelerator physics problems in an efficient manner, there are still ample opportunities for future improvements toward using BO in accelerator science.

First and foremost is continuing research in the integration of physics information into GP models. As has been highlighted in several sections of this review, improving the accuracy of GP modeling improves decision making during optimization, leading to faster convergence to optimal solutions and reductions in the number of constraint violations. Incorporating information into GP models before performing optimization is especially critical in making good decisions during the first few iterations. Furthermore, if uncertainties exist in the sources of information used, these uncertainties should be incorporated into the GP model as well.

In both online accelerator operations and simulated optimization, improving the orchestration of objective function evaluation, GP model generation, and acquisition function maximization is another source of major potential improvements. The development of a centralized control framework that dispatches these tasks contained in BO on parallel resources could lead to major reductions in the overall cost of performing optimization. A potential example of this would be an online accelerator control program that would send current and/or future potential machine states to be evaluated on high-performance computing clusters outside the control room. Results collected from these physics simulations could be used to inform online control in real time, similar to what is done in [156].

VII. CONCLUSION

In conclusion, BO algorithms are an effective, extendable way of solving a wide variety of optimization challenges in accelerator physics. BO algorithms are particularly well suited for addressing optimization challenges that involve objectives and constraints that require significant beam time, personnel, or computational resources to evaluate. These algorithms use statistical surrogate models based on gathered data to inform optimization, reducing the number of objective function evaluations versus other black-box optimization schemes. As a result, the BO framework provides a straightforward and robust way to incorporate prior knowledge (either from past measurements or physics information) or approximate measurements/computation into the modeling process to further improve optimization convergence speed. By modifying standard acquisition functions, BO algorithms can be customized to solve a wide variety of single, multiobjective, and characterization problems in accelerator physics. Used correctly, BO algorithms can reduce the overall cost of performing optimization when compared to conventional black box optimization algorithms, allowing accelerator scientists to address more complex optimization challenges provided that they effectively balance the costs of evaluating objectives and constraints with the costs of algorithmic decision making.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515. This manuscript acknowledges the support of Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. This work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357, and through a "Data, Artificial Intelligence, and Machine Learning at DOE Scientific User Facilities" Grant from the DOE's Office of Nuclear Physics supporting the ATLAS accelerator facility. This work was supported in part by the U.S. National Science Foundation under Award No. PHY-1549132, the Center for Bright Beams". This work has in part been funded by the IVF project InternLabs-0011 (HIR3X) and the Initiative and Networking Fund by the Helmholtz Association (Autonomous Accelerator, ZT-I-PF-5-6). The authors acknowledge support from DESY (Hamburg, Germany) and KIT (Karlsruhe, Germany), members of the Helmholtz Association HGF. Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy. M. J. V. S. acknowledges support from the Royal Society URF-R1221874. W. L. acknowledges support from the U.S. National Science Foundation under Award No. PHY-1549132. This project has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 871072. This work was in part supported by the CAMPA collaboration, a project of the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program.

Conceptualization, R. R., A. S. G., and A. L. E.; Data curation, R. R.; Visualization, R. R., T. B., M. J. V. S., J. O. L., N. K., A. L. E., S. M. L, R. L.; Writing—original draft, R. R., D. K., Y. G., W. L., T. B., C. X., A. S. G, J. M., J. K., A. E., J. O. L., N. K., V. K., W. N., Z. Z., R. L., and A. L. E.; Writing—review and editing, R. R., D. K., W. N., N. M. I, Y. G., W. L., T. B., M. J. V. S., A. E., J. K., C. X., A. S. G., B. M., N. K., V. K., X. H., D. R., J. S. J, and A. L. E. All authors have read and agreed to the published version of the manuscript.

[1] U.S. Particle Physics Community Planning Exercise. "2023 P5 Report: Exploring the Quantum Universe." U.S. Particle Physics, 2023, https://www.usparticlephysics.org/2023-p5-report/.

[2] U.S. Department of Energy, Office of Science. "Accelerator and Beam Physics Roadmap 2023". 2023. U.S. Department of Energy, www.science.osti.gov/hep/-/media/hep/pdf/2022/ABP_Roadmap_2023_final.pdf.

[3] J. Mockus, *The Bayesian Approach to Local Optimization* (Springer, Netherlands, 1989), pp. 125–156.

[4] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, Taking the human out of the loop: A review of Bayesian optimization, Proc. IEEE **104**, 148 (2016).

[5] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches* (Springer Science and Business Media, Heidelberg, 2013).

[6] I. V. Bazarov and C. K. Sinclair, Multivariate optimization of a high brightness dc gun photoinjector, Phys. Rev. ST Accel. Beams **8**, 034202 (2005).

[7] J. P. Coutinho, L. O. Santos, and M. S. Reis, Bayesian optimization for automatic tuning of digital multi-loop PID controllers, Comput. Chem. Eng. **173**, 108211 (2023).

[8] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, MA, 2006).

[9] M. McIntire, T. Cope, S. Ermon, and D. Ratner, Bayesian optimization of FEL performance at LCLS, in *Proceedings of the 7th International Particle Accelerator Conference, IPAC-2016, Busan, Korea* (JACoW, Geneva, Switzerland, 2016), WEPOW055.

[10] J. Duris, D. Kennedy, A. Hanuka, J. Shtalenkova, A. Edelen, P. Baxevanis, A. Egger, T. Cope, M. McIntire, S. Ermon, and D. Ratner, Bayesian optimization of a free-electron laser, Phys. Rev. Lett. **124**, 124801 (2020).

[11] C. Xu, T. Boltz, A. Mochihashi, A. Santamaria Garcia, M. Schuh, and A.-S. Müller, Bayesian optimization of the beam injection process into a storage ring, Phys. Rev. Accel. Beams **26**, 034601 (2023).

[12] S. Miskovich, F. Montes, G. Berg, J. Blackmon, K. Chipps, M. Couder, C. Deibel, K. Hermansen, A. Hood, R. Jain, T. Ruland, H. Schatz, M. Smith, P. Tsintari, and L. Wagner, Online Bayesian optimization for a recoil mass separator, Phys. Rev. Accel. Beams **25**, 044601 (2022).

[13] E. Iwai, I. Inoue, H. Maesaka, T. Inagaki, M. Yabashi, T. Hara, and H. Tanaka, Spectral-brightness optimization of an x-ray free-electron laser by machine-learning-based tuning, J. Synchrotron Radiat. **30**, 1048 (2023).

[14] Y. Morita, T. Washio, and Y. Nakashima, Accelerator tuning method using autoencoder and Bayesian optimization, Nucl. Instrum. Methods Phys. Res., Sect. A **1057**, 168730 (2023).

[15] A. Awal, J. Hetzel, R. Gebel, V. Kamerdzhiev, and J. Pretz, Optimization of the injection beam line at the cooler synchrotron COSY using Bayesian optimization, J. Instrum. **18**, P04010 (2023).

[16] Y. Gao, W. Lin, K. A. Brown, X. Gu, G. H. Hoffstaetter, J. Morris, and S. Seletskiy, Bayesian optimization experiment for trajectory alignment at the low energy RHIC electron cooling system, Phys. Rev. Accel. Beams **25**, 014601 (2022).

[17] E. Salehi and M. Katoh, Bayesian optimization of the dynamic aperture in UVSOR-IV design study, J. Phys. Conf. Ser. **2687**, 032030 (2024).

[18] R. J. Shalloo *et al.*, Automation and control of laser wakefield accelerators using Bayesian optimization, Nat. Commun. **11**, 6355 (2020).

[19] S. Jalas, M. Kirchen, P. Messner, P. Winkler, L. Hübner, J. Dirkwinkel, M. Schnepp, R. Lehe, and A. R. Maier, Bayesian Optimization of a Laser-Plasma Accelerator, Phys. Rev. Lett. **126**, 104801 (2021).

[20] H. Ye, Y. Gu, X. Zhang, S. Wang, F. Tan, J. Zhang, Y. Yang, Y. Yan, Y. Wu, W. Huang, and W. Zhou, Fast optimization for betatron radiation from laser wakefield acceleration based on Bayesian optimization, Results Phys. **43**, 106116 (2022).

[21] B. Loughran *et al.*, Automated control and optimisation of laser driven ion acceleration, High Power Laser Sci. Eng. **11**, e35 (2023).

[22] S. Jalas, M. Kirchen, C. Braun, T. Eichner, J. B. Gonzalez, L. Hübner, T. Hülsenbusch, P. Messner, G. Palmer, M. Schnepp, C. Werle, P. Winkler, W. P. Leemans, and A. R. Maier, Tuning curves for a laser-plasma accelerator, Phys. Rev. Accel. Beams **26**, 071302 (2023).

[23] R. Sha, B. Wang, J. Zhao, X.-J. Duan, L. Yan, and T. Yu, Bayesian optimization for design of high-repetition-rate laser-driven muon source, Front. Phys. **11**, 1233733 (2023).

[24] T. Boltz, J. L. Martinez, C. Xu, K. R. Baker, R. Roussel, D. Ratner, B. Mustapha, and A. L. Edelen, More sample-efficient tuning of particle accelerators with Bayesian optimization and prior mean models, arXiv:2403.03225.

[25] K. Hwang, K. Fukushima, T. Maruta, S. Nash, P. Ostroumov, A. Plastun, T. Zhang, and Q. Zhao, Beam tuning at the FRIB front end using machine learning, in *Proceedings of the 13th Particle Accelerator Conference, IPAC-2022, Bangkok, Thailand* (JACoW, Geneva, Switzerland, 2022), pp. 983–986.

[26] N. Kuklev, M. Borland, G. I. Fystro, H. Shang, and Y. Sun, Online accelerator tuning with adaptive Bayesian optimization, in *Proceedings of the 5th North American Particle Accelerator Conference, NAPAC'22, Albuquerque, NM* (JACoW, Geneva, Switzerland, 2022), p. 842.

[27] N. Kuklev, Y. Sun, H. Shang, M. Borland, and G. I. Fystro, Robust adaptive Bayesian optimization, in *Proceedings of the 14th International Particle Accelerator Conference, IPAC-2023, Venice, Italy* (JACoW, Geneva, Switzerland, 2023), pp. 4377–4380.

[28] R. Roussel, A. Edelen, D. Ratner, K. Dubey, J. P. Gonzalez-Aguilera, Y. K. Kim, and N. Kuklev, Differentiable preisach modeling for characterization and optimization of particle accelerator systems with hysteresis, Phys. Rev. Lett. **128**, 204801 (2022).

[29] R. Roussel, J. P. Gonzalez-Aguilera, Y.-K. Kim, E. Wisniewski, W. Liu, P. Piot, J. Power, A. Hanuka, and A. Edelen, Turn-key constrained parameter space exploration for particle accelerators using Bayesian active learning, Nat. Commun. **12**, 5612 (2021).

[30] J. Kirschner, M. Mutny, N. Hiller, R. Ischebeck, and A. Krause, Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces, in *Proceedings of the 36th International Conference on Machine Learning* (PMLR, 2019), pp. 3429–3438.

[31] J. Kirschner, M. Mutný, A. Krause, J. C. de Portugal, N. Hiller, and J. Snuverink, Tuning particle accelerators with safety constraints using bayesian optimization, Phys. Rev. Accel. Beams **25**, 062802 (2022).

[32] J. Luebsen, M. Schuette, S. Schulz, and A. Eichler, A safe Bayesian optimization algorithm for tuning the optical synchronization system at european XFEL, IFACPapersOnLine, **56**, 3079 (2023).

[33] F. Ji, A. Edelen, R. Roussel, X. Shen, S. Miskovich, S. Weathersby, D. Luo, M. Mo, P. Kramer, C. Mayes, M. A. K. Othman, E. Nanni, X. Wang, A. Reid, M. Minitti, and R. J. England, Multi-objective Bayesian active learning for MeV-ultrafast electron diffraction, Nat. Commun. **15**, 4726 (2024).

[34] R. Roussel, A. Hanuka, and A. Edelen, Multiobjective Bayesian optimization for online accelerator tuning, Phys. Rev. Accel. Beams **24**, 062801 (2021).

[35] F. Irshad, S. Karsch, and A. Döpp, Multi-objective and multi-fidelity Bayesian optimization of laser-plasma acceleration, Phys. Rev. Res. **5**, 013063 (2023).

[36] X. Huang, M. Song, and Z. Zhang, Multi-objective multi-generation Gaussian process optimizer, in *Proc. IPAC'21, Campinas, Brazil, 2021* (2021), pp. 3383–3386, 10.18429/JACoW-IPAC2021-WEPAB304.

[37] S. A. Miskovich, W. Neiswanger, W. Colocho, C. Emma, J. Garrahan, T. Maxwell, C. Mayes, S. Ermon, A. Edelen, and D. Ratner, Multipoint-bax: A new approach for efficiently tuning particle accelerator emittance via virtual objectives, Mach. Learn. **5**, 015004 (2024).

[38] A. Ferran Pousa, S. Jalas, M. Kirchen, A. Martinez de la Ossa, M. Thévenet, S. Hudson, J. Larson, A. Huebl, J.-L. Vay, and R. Lehe, Bayesian optimization of laser-plasma accelerators assisted by reduced physical models, Phys. Rev. Accel. Beams **26**, 084601 (2023).

[39] M. Krasser and DodoTheDeveloper, krasserm/Bayesian-machine-learning: v-0.3 (2020).

[40] J.-W. van de Meent *et al.*, An introduction to probabilistic programming, arXiv:1809.10756.

[41] The posterior discussed here is often referred to as the *posterior predictive*. The true posterior of the GP model is with respect to function values and is written as $p(\mathbf{f}_*|\mathbf{X}_*, \mathcal{D})$.

[42] R. M. Neal, *Bayesian Learning for Neural Networks* (Springer Science and Business Media, New York, 2012), Vol. 118.

[43] M. G. Genton, Classes of kernels for machine learning: A statistics perspective, J. Mach. Learn. Res. **2**, 299 (2002).

[44] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, Efficient BackProp, in *Neural Networks: Tricks of the Trade: Second Edition*, edited by G. Montavon, G. B. Orr, and K.-R. Müller, Lecture Notes in Computer Science (Springer, Berlin, Heidelberg, 2012), pp. 9–48.

[45] G. H. Golub, P. C. Hansen, and D. P. O'Leary, Tikhonov regularization and total least squares, SIAM J. Matrix Anal. Appl. **21**, 185 (1999).

[46] C. J. Geyer, Practical Markov chain Monte Carlo, Stat. Sci. **7**, 473 (1992).

[47] D. Eriksson and M. Jankowiak, High-dimensional Bayesian optimization with sparse axis-aligned subspaces, in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021* (PMLR, 2021), pp. 493–503.

[48] A. Hanuka, X. Huang, J. Shtalenkova, D. Kennedy, A. Edelen, Z. Zhang, V. R. Lalchand, D. Ratner, and J. Duris, Physics model-informed Gaussian process for online optimization of particle accelerators, Phys. Rev. Accel. Beams **24**, 072802 (2021).

[49] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, Deep kernel learning, in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (PMLR, 2016), pp. 370–378.

[50] C. Xu, R. Roussel, and A. Edelen, Neural network prior mean for particle accelerator injector tuning, arXiv:2211.09028.

[51] K. Hwang, T. Maruta, A. Plastun, K. Fukushima, T. Zhang, Q. Zhao, P. Ostroumov, and Y. Hao, Prior-mean-assisted Bayesian optimization application on FRIB Front-End tuning, arXiv:2211.06400.

[52] J. Kaiser, C. Xu, A. Eichler, and A. Santamaria Garcia, Bridging the gap between machine learning and particle accelerator physics with high-speed, differentiable simulations, Phys. Rev. Accel. Beams **27**, 054601 (2024).

[53] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Netw. **2**, 359 (1989).

[54] A. Edelen, N. Neveu, M. Frey, Y. Huber, C. Mayes, and A. Adelmann, Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems, Phys. Rev. Accel. Beams **23**, 044601 (2020).

[55] D. Eriksson and M. Poloczek, Scalable constrained Bayesian optimization, in *International Conference on Artificial Intelligence and Statistics* (PMLR, 2021), pp. 730–738.

[56] A. G. Wilson and Z. Ghahramani, Copula processes, Adv. Neural Inf. Process. Syst. **23**, 2460 (2010), https://proceedings.neurips.cc/paper_files/paper/2010/file/fc8001f834f6a5f0561080d134d53d29-Paper.pdf.

[57] F. M. Nyikosa, M. A. Osborne, and S. J. Roberts, Bayesian optimization for dynamic problems, arXiv:1803.03432.

[58] A. Krause and C. Ong, Contextual Gaussian process bandit optimization, in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, Vol. 24, edited by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger (Curran Associates, Inc., Red Hook, NY, 2011).

[59] A. Wilson and R. Adams, Gaussian process kernels for pattern discovery and extrapolation, in *Proceedings of the 30th International Conference on Machine Learning*, edited by S. Dasgupta and D. McAllester, Proceedings of Machine Learning Research Vol. 28 (PMLR, 2013), pp. 1067–1075.

[60] K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos, Multi-fidelity Bayesian optimisation with continuous approximations, in *Proceedings of the 34th International Conference on Machine Learning, ICML-2017, Sydney, New South Wales, Australia* (JMLR.org, 2017), Vol. 70, pp. 1799–1808.

[61] E. V. Bonilla, K. Chai, and C. Williams, Multi-task Gaussian process prediction, in *Proceedings of the 20th International Conference on Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, NY, 2007), Vol. 20.

[62] F. Preisach, Über die magnetische Nachwirkung, Z. Phys. **94**, 277 (1935).

[63] E. Brochu, V. M. Cora, and N. de Freitas, A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, arXiv:1012.2599.

[64] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, Gaussian process optimization in the Bandit setting: No regret and experimental design, in *Proceedings of the*

27th International Conference on Machine Learning (ICML 2010) (Omnipress, Madison, WI, 2010), p. 1015.

[65] S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy, Unexpected improvements to expected improvement for bayesian optimization, arXiv:2310.20708.

[66] J. T. Wilson, R. Moriconi, F. Hutter *et al.*, The reparameterization trick for acquisition functions, arXiv:1712.00424v1.

[67] M. Noack, K. G. Yager, M. Fukuto, G. S. Doerk, R. Li, and J. Sethian, A Kriging-based approach to autonomous experimentation with applications to X-ray scattering, Sci. Rep. **9**, 1 (2019).

[68] M. M. Noack, P. H. Zwart, D. M. Ushizima, M. Fukuto, K. G. Yager, K. C. Elbert, C. B. Murray, A. Stein, G. S. Doerk, E. H. Tsai *et al.*, Gaussian processes for autonomous data acquisition at large-scale synchrotron and neutron facilities, Nat. Rev. Phys. **3**, 685 (2021).

[69] M. M. Noack, G. S. Doerk, R. Li, J. K. Streit, R. A. Vaia, K. G. Yager, and M. Fukuto, Autonomous materials discovery driven by Gaussian process regression with inhomogeneous measurement noise and anisotropic kernels, Sci. Rep. **10**, 17663 (2020).

[70] S. P. Boyd and L. Vandenberghe, *Convex Optimization* (Cambridge University Press, Cambridge, England, 2004).

[71] F. Berkenkamp, A. Krause, and A. P. Schoellig, Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics, Mach. Learn. **112**, 3713 (2021).

[72] Y. Kim, R. Allmendinger, and M. López-Ibáñez, Safe learning and optimization techniques: Towards a survey of the state of the art, in *Proceedings of the International Workshop on the Foundations of Trustworthy AI Integrating Learning, Optimization and Reasoning* (Springer, New York, 2020), pp. 123–139.

[73] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, Bayesian optimization with inequality constraints, in *Proceedings of the 31st International Conference on Machine Learning, Beijing, China, ICML-2014* (JMLR.org, 2014), Vol. 32, pp. 937–945.

[74] R. Roussel, D. Kennedy, A. Edelen, S. Kim, E. Wisniewski, and J. Power, Demonstration of autonomous emittance characterization at the Argonne Wakefield Accelerator, Instruments **7**, 29 (2023).

[75] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, Safe exploration for optimization with Gaussian processes, in *Proceedings of the 32nd International Conference on Machine Learning* (PMLR, 2015), Vol. 37, pp. 997–1005.

[76] H. H. Sohrab, *Basic Real Analysis* (Springer, New York, 2003), Vol. 231.

[77] Y. Sui, V. Zhuang, J. Burdick, and Y. Yue, Stagewise safe Bayesian optimization with Gaussian processes, in *Proceedings of the 35th International on Conference on Machine Learning, ICML-2018* (PMLR, 2018), pp. 4781–4789.

[78] M. Turchetta, F. Berkenkamp, and A. Krause, Safe exploration for interactive machine learning, in Proceedings of the 33rd International Conference on Neural Information Processing Systems (Curran Associates Inc., Red Hook, NY, 2019), Vol. 32, pp. 2891–2901.

[79] M. Song, X. Huang, L. Spentzouris, and Z. Zhang, Storage ring nonlinear dynamics optimization with multi-objective multi-generation Gaussian process optimizer, Nucl. Instrum. Methods Phys. Res., Sect. A **976**, 164273 (2020).

[80] H. R. Maier, S. Razavi, Z. Kapelan, L. S. Matott, J. Kasprzyk, and B. A. Tolson, Introductory overview: Optimization using evolutionary algorithms and other metaheuristics, Environ. Modell. Software **114**, 195 (2019).

[81] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. **6**, 182 (2002).

[82] S. Daulton, M. Balandat, and E. Bakshy, Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization, Adv. Neural Inf. Process. Syst. **33**, 9851 (2020), https://proceedings.neurips.cc/paper/2020/file/6fec24eac8f18ed793f5eaad3dd7977c-Paper.pdf.

[83] F. Ji, A. Edelen, R. England, P. Kramer, D. Luo, C. Mayes, M. Minitti, S. Miskovich, M. Mo, A. Reid, R. Roussel, X. Shen, X. Wang, and S. Weathersby, Multi-objective Bayesian optimization at SLAC MeV-UED, in *Proceedings of the 13th International Particle Accelerator Conference, IPAC-2022, Bangkok, Thailand* (JACoW, Geneva, Switzerland, 2022), pp. 995–998.

[84] R. P. Liem, G. K. W. Kenway, and J. R. R. A. Martins, Multimission aircraft fuel-burn minimization via multipoint aerostructural optimization, AIAA J. **53**, 104 (2015).

[85] W. Neiswanger, K. A. Wang, and S. Ermon, Bayesian algorithm execution: Estimating computable properties of black-box functions using mutual information, in *Proceedings of the International Conference on Machine Learning* (PMLR, 2021), pp. 8005–8015.

[86] R. Roussel and A. Edelen, Proximal biasing for Bayesian optimization and characterization of physical systems, in *Proceedings of Fourth Workshop on Machine Learning and the Physical Sciences* (NeurIPS, 2021).

[87] B. Letham and E. Bakshy, Bayesian optimization for policy search via online-offline experimentation, arXiv:1904.01049.

[88] J. Wu and P. I. Frazier, Continuous-fidelity Bayesian optimization with knowledge gradient, in *Proceedings of the 31st Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA* (Curran Associates Inc., Red Hook, NY, 2017).

[89] F. Irshad, S. Karsch, and A. Döpp, Leveraging trust for joint multi-objective and multi-fidelity optimization, arXiv:2112.13901.

[90] J. Wilson, F. Hutter, and M. Deisenroth, Maximizing acquisition functions for Bayesian optimization, in *Proceedings of the 32nd Conference on Neural Information Processing Systems, NeurIPS 2018, Montréal, Canada* (Curran Associates Inc., Red Hook, NY, 2017), Vol. 31.

[91] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[92] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. **16**, 1190 (1995).

[93] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, NY, 2019), pp. 8026–8037.

[94] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, Scalable global optimization via local Bayesian optimization, in *Proceedings of 33rd Conference on Neural Information Processing Systems, NeurIPS 2019, Vancouver, Canada* (Curran Associates Inc., Red Hook, NY, 2019), Vol. 32.

[95] P. Raimondi *et al.*, The extremely brilliant source storage ring of the European Synchrotron Radiation Facility, Commun. Phys. **6**, 82 (2023).

[96] S. M. Liuzzo *et al.*, Optimisation of the Touschek lifetime in Synchrotron Light Sources using badger, in *Proceedings of the 19th International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS-2023, Cape Town, South Africa* (JACoW, Geneva, Switzerland, 2023), MO3AO01.

[97] D. Ginsbourger, R. Le Riche, and L. Carraro, Kriging is well-suited to parallelize optimization, in *Computational Intelligence in Expensive Optimization Problems* (Springer, New York, 2010), pp. 131–162.

[98] D. J. Rezende, S. Mohamed, and D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in *Proceedings of the 31st International Conference on Machine Learning, Beijing, China* (PMLR, 2014).

[99] M. Balandat, B. Karrer, D. Jiang *et al.*, BoTorch: A framework for efficient Monte-Carlo Bayesian optimization, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, NY, 2020), Vol. 33.

[100] J. Gonzalez, Z. Dai, P. Hennig, and N. Lawrence, Batch Bayesian optimization via local penalization, in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, Cadiz, Spain*, edited by A. Gretton and C. C. Robert (JMLR W&CP, 2016), Vol. 51, pp. 648–657.

[101] C. Xu, E. Bründermann, A.-S. Müller, A. Santamaria Garcia, M. Schwarz, and J. Schäfer, Optimization studies of simulated THz radiation at FLUTE, in *Proceedings of the 13th International Particle Accelerator Conference, IPAC-2022, Bangkok, Thailand* (JACoW, Geneva, Switzerland, 2022), pp. 2292–2295.

[102] Z. Zhang, M. Song, and X. Huang, Online accelerator optimization with a machine learning-based stochastic algorithm, Mach. Learn. **2**, 015014 (2020).

[103] K. Liagkouras and K. Metaxiotis, An elitist polynomial mutation operator for improved performance of MOEAs in computer networks, in *Proceedings of 2013 22nd International Conference on Computer Communication and Networks ICCCN, Nassau, Bahamas* (IEEE, New York, 2013), pp. 1–5.

[104] K. Deb and R. B. Agrawal, Simulated binary crossover for continuous search space, Complex Syst. **9**, 115 (1995).

[105] L. Emery, H. Shang, Y. Sun, and X. Huang, Application of a machine learning based algorithm to online optimization of the nonlinear beam dynamics of the Argonne Advanced Photon Source, Phys. Rev. Accel. Beams **24**, 082802 (2021).

[106] H. Shang, Y. Sun, X. Huang, M. Borland, M. Song, and Z. Zhang, Experience with on-line optimizers for APS linac front end optimization, in *Proceedings of the 12th International Particle Accelerator Conference, IPAC-2021, Campinas, SP, Brazil* (JACoW, Geneva, Switzerland, 2021), pp. 2151–2154.

[107] F. Lohl (private communication).

[108] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016), http://www.deeplearningbook.org.

[109] S. Ruder, An overview of gradient descent optimization algorithms, arXiv:1609.04747.

[110] T. Dorigo, A. Giammanco, P. Vischia, M. Aehle, M. Bawaj, A. Boldyrev, P. de Castro Manzano, D. Derkach, J. Donini, A. Edelen *et al.*, Toward the end-to-end optimization of particle physics instruments with differentiable programming, Rev. Phys. **10**, 100085 (2023).

[111] D. Ratner, F. Christie, J. Cryan, A. Edelen, A. Lutman, and X. Zhang, Recovering the phase and amplitude of x-ray fel pulses using neural networks and differentiable models, Opt. Express **29**, 20336 (2021).

[112] N. Kuklev, Differentiable beam optics optimization and measurement, in *Proceedings of the 14th International Particle Accelerator Conference, IPAC-2023, Bangkok, Thailand* (JACoW, Geneva, Switzerland, 2023), pp. 3108–3111.

[113] J. P. Gonzalez-Aguilera, Y.-K. Kim, R. Roussel, A. Edelen, and C. Mayes, Towards fully differentiable accelerator modeling, in *Proc. IPAC'23, Venice, Italy* (2023), pp. 2797–2800, 10.18429/JACoW-IPAC2023-WEPA065.

[114] R. Roussel, A. Edelen, C. Mayes, D. Ratner, J. P. Gonzalez-Aguilera, S. Kim, E. Wisniewski, and J. Power, Phase space reconstruction from accelerator beam measurements using neural networks and differentiable simulations, Phys. Rev. Lett. **130**, 145001 (2023).

[115] O. Stein, J. Kaiser, and A. Eichler, Accelerating linear beam dynamics simulations for machine learning applications, in *Proceedings of the 13th International Particle Accelerator Conference, Bangkok, Thailand* (JACoW, Geneva, Switzerland, 2022).

[116] A. Edelen, J. Edelen, S. Biedron, S. Milton, and P. van der Slot, Using a neural network control policy for rapid switching between beam parameters in a FEL, in *Proceedings of the 38th International Free Electron Laser Conference, FEL2017, Santa Fe, NM* (JACoW, Geneva, Switzerland, 2017), WEP031, 10.18429/JACoW-FEL2017-WEP031.

[117] J. A. Nelder and R. Mead, A simplex method for function minimization, Comput. J. **7**, 308 (1965).

[118] L. Emery, M. Borland, and H. Shang, Use of a general-purpose optimization module in accelerator control, in *Proceedings of the 20th Particle Accelerator Conference, PAC-2003, Portland, OR* (IEEE, New York, 2003), Vol. 4, pp. 2330–2332.

[119] Z. Zhang, A. Edelen, C. Mayes, J. Garrahan, J. Shtalenkova, R. multi, S. Miskovich, D. Ratner, M. Boese, S. Tomin, G. Wang, and Y. Hidaka, Badger: The missing optimizer in ACR, in *Proc. IPAC'22, Bangkok, Thailand, 2022* (2022), pp. 999–1002, 10.18429/JACoW-IPAC2022-TUPOST058.

[120] I. Agapov, S. Tomin, W. Decking, M. Scholz, I. Zagorodnov, and G. Geloni, *On-line optimization of European XFEL with ocelot* (Deutsches Elektronen-Synchrotron, DESY, Hamburg, 2017).

[121] H. Shang and M. Borland, A parallel simplex optimizer and its application to high-brightness storage ring design, in *Proceedings of the 21st Particle Accelerator Conference, Knoxville, TN, 2005* (IEEE, Piscataway, NJ, 2005), pp. 4230–4232.

[122] X. Huang, Robust simplex algorithm for online optimization, Phys. Rev. Accel. Beams **21**, 104601 (2018).

[123] J. Kaiser, C. Xu, A. Eichler, A. Santamaria Garcia, O. Stein, E. Bründermann, W. Kuropka, H. Dinter, F. Mayet, T. Vinatier, F. Burkart, and H. Schlarb, Reinforcement learning-trained optimisers and Bayesian optimisation for online particle accelerator tuning, Sci. Rep. **14**, 1 (2024).

[124] C. X. *et al.*, Bayesian optimization for SASE tuning at the European XFEL, in *Proceedings of the 14th International Particle Accelerator Conference, IPAC-2023, Venice, Italy* (JACoW, Geneva, Switzerland, 2023), pp. 4483–4486.

[125] E. Chong and S. Zak, *An Introduction to Optimization*, 4th ed. (John Wiley and Sons, Inc., New York, 2013).

[126] X. Huang, J. Corbett, J. Safranek, and J. Wu, An algorithm for online optimization of accelerators, Nucl. Instrum. Methods Phys. Res., Sect. A **726**, 77 (2013).

[127] Z. Zhang, M. Song, and X. Huang, Optimization method to compensate accelerator performance drifts, Phys. Rev. Accel. Beams **25**, 122801 (2022).

[128] M. J. Powell *et al.*, The BOBYQA algorithm for bound constrained optimization without derivatives, University of Cambridge, Cambridge, Report No. DAMTP 2009/NA06, 2009.

[129] N. Neveu, J. Larson, J. Power, and L. Spentzouris, Photoinjector optimization using a derivative-free, model-based trust-region algorithm for the Argonne Wakefield Accelerator, J. Phys. Conf. Ser. **874**, 012062 (2017).

[130] S. Appel and S. Reimann, Beam line optimization using derivative-free algorithms, J. Phys. Conf. Ser. **1350**, 012104 (2019).

[131] A. Scheinker, S. Hirlaender, F. M. Velotti, S. Gessner, G. Z. D. Porta, V. Kain, B. Goddard, and R. Ramjiawan, Online multi-objective particle accelerator optimization of the awake electron beam line for simultaneous emittance and orbit control, AIP Adv. **10**, 055320 (2020).

[132] A. Scheinker, X. Pang, and L. Rybarcyk, Model-independent particle accelerator tuning, Phys. Rev. ST Accel. Beams **16**, 102803 (2013).

[133] A. Scheinker, E.-C. Huang, and C. Taylor, Extremum seeking-based control system for particle accelerator beam loss minimization, IEEE Trans. Control Syst. Technol. **30**, 2261 (2022).

[134] A. Scheinker, A. Edelen, D. Bohler, C. Emma, and A. Lutman, Demonstration of model-independent control of the longitudinal phase space of electron beams in the Linac-Coherent Light Source with femtosecond resolution, Phys. Rev. Lett. **121**, 044801 (2018).

[135] J. Kaiser, O. Stein, and A. Eichler, Learning-based optimisation of particle accelerators under partial observability without real-world training, in *Proceedings of the 39th International Conference on Machine Learning*, edited by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato (PMLR, 2022), Vol. 162, pp. 10575–10585.

[136] V. Kain, S. Hirlander, B. Goddard, F. M. Velotti, G. Z. Della Porta, N. Bruchon, and G. Valentino, Sample-efficient reinforcement learning for cern accelerator control, Phys. Rev. Accel. Beams **23**, 124801 (2020).

[137] D. Meier, L. V. Ramirez, J. Völker, J. Viefhaus, B. Sick, and G. Hartmann, Optimizing a superconducting radio-frequency gun using deep reinforcement learning, Phys. Rev. Accel. Beams **25**, 104604 (2022).

[138] T. Boltz *et al.*, Feedback design for control of the micro-bunching instability based on reinforcement learning, in *Proceedings of the IPAC'19* (JACoW, Geneva, Switzerland, 2019), pp. 104–107.

[139] D. L. Kafkes and M. Schram, Developing robust digital twins and reinforcement learning for accelerator control systems at the fermilab booster, in *Proceedings of the IPAC'21* (JACoW Publishing, Geneva, Switzerland, 2021), pp. 2268–2271.

[140] K. Li and J. Malik, Learning to optimize, arXiv:1606.01885.

[141] E. F. Camacho and C. Bordons, Introduction to model predictive control, in *Model Predictive Control* (Springer London, London, 2007), pp. 1–11.

[142] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. (MIT Press, Cambridge, MA, 2018).

[143] R. A. Howard, *Dynamic Programming and Markov Processes* (MIT Press, Cambridge, MA, 1960).

[144] S. C. Leemann, S. Liu, A. Hexemer, M. A. Marcus, C. N. Melton, H. Nishimura, and C. Sun, Demonstration of machine learning-based model-independent stabilization of source properties in synchrotron light sources, Phys. Rev. Lett. **123**, 194801 (2019).

[145] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, Stochastic variational deep kernel learning, arXiv:1611.00336.

[146] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter, Bayesian optimization with robust Bayesian neural networks, in *Advances in Neural Information Processing Systems*, edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., Red Hook, NY, 2016), Vol. 29.

[147] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. **12**, 2825 (2011).

[148] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, BoTorch: A framework for efficient Monte-Carlo Bayesian optimization, in *Advances in Neural Information Processing Systems 33* (Curran Associates Inc., Red Hook, NY, 2020).

[149] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration, in *Proceedings of 32nd Conference on Neural Information Processing Systems NeurIPS 2018, Montréal, Canada* (Curran Associates Inc., Red Hook, NY, 2018).

[150] R. Roussel, A. Edelen, A. Bartnik, and C. Mayes, Xopt: A simplified framework for optimization of accelerator problems using advanced algorithms, in *Proceedings of the 14th International Particle Accelerator Conference, IPAC-2023, Venice, Italy* (JACoW, Geneva, Switzerland, 2023), pp. 4796–4799.

[151] Ryan Roussel, Christopher Mayes, Nikita Kuklev, Tobias Boltz, Jacqueline Garrahan, zhe-slac, dylanmkennedy, YektaY, Christopher M. Pierce, Hugo Slepicka, kathryn-baker, Ken Lauer, and Zhe Zhang, Christopher-Mayes/Xopt: Xopt v2.3.0 (v2.3.0). Zenodo (2024), 10.5281/zenodo.12168937

[152] Z. Zhang, Badger: The Ocelot Optimizer Rebirth, 10.11578/dc.20220706.3.

[153] S. Hudson, J. Larson, J.-L. Navarro, and S. M. Wild, libensemble: A library to coordinate the concurrent evaluation of dynamic ensembles of calculations, IEEE Trans. Parallel Distrib. Syst. **33**, 977 (2022).

[154] N. Kuklev *et al.*, ML-enhanced commissioning of the APS-U accelerator complex, in *IPAC'24, Nashville, TN, USA, 2024* (unpublished), paper TUPS50.

[155] N. Madysa and V. Kain, The Generic Optimization Framework and Frontend (0.13.0). Zenodo. Geneva, Switzerland, 10.5281/zenodo.8434513.

[156] C. Gulliford, A. Bartnik, I. Bazarov, L. Cultrera, J. Dobbins, B. Dunham, F. Gonzalez, S. Karkare, H. Lee, H. Li *et al.*, Demonstration of low emittance in the Cornell energy recovery linac injector prototype, Phys. Rev. ST Accel. Beams **16**, 073401 (2013).