ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WESV001

XOPT AND BADGER: A MACHINE LEARNING ECOSYSTEM FOR REAL-TIME ACCELERATOR CONTROL AND OPTIMIZATION

R. Roussel* D. Kennedy, A. Edelen, S. Miskovich, Z. Zhang,
SLAC National Accelerator Laboratory, Menlo Park, CA, USA
N. Kuklev, Fermi National Accelerator Laboratory, Batavia, IL, USA

Abstract

Machine learning (ML)-based black-box optimization algorithms have demonstrated significant improvements in accelerator optimization speed, often by orders of magnitude. However, deploying these algorithms in real-time facility control remains challenging due to the specialized expertise and infrastructure required. To bridge this gap, we introduce the Xopt ecosystem, a versatile suite of tools designed to make advanced ML-based optimization accessible to the broader accelerator community. This ecosystem includes Xopt, a modular Python framework that facilitates the integration of ML-based optimization algorithms with arbitrary control problems, and Badger, a graphical user interface built on top of Xopt, which enables seamless deployment of ML algorithms in real-time control systems. The Xopt ecosystem has been successfully applied towards solving challenging real-time control problems at leading international accelerator facilities, including SLAC, LBNL, Argonne, Fermilab, BNL, DESY, and ESRF, demonstrating its effectiveness in real-world optimization tasks. In this presentation, we provide an overview of Xopt's capabilities and illustrate its impact through case studies from SLAC accelerator facilities including LCLS, LCLS-II, and FACET-II.

INTRODUCTION

The need to solve complex optimization problems is widely prevalent in the field of accelerator physics. For example, accelerator control parameters must be tuned during accelerator operations to improve performance (so-called "online" tuning) or accelerator design parameters must be optimized in simulation to implement novel operational modes ("offline tuning"). These problems can be solved using a wide variety of conventional optimization algorithms, including Nelder-Mead Simplex [1] and Robust Conjugate Direction Search [2]. In addition, evolutionary algorithms, such as NSGA-II [3], can leverage the power of parallelized computation to solve multi-objective optimization problems. More recently, advanced machine learning based algorithms have been used to solve both online and offline accelerator optimization problems, most notably Bayesian Optimization (BO) [4].

Despite their wide applicability, connecting these algorithms to optimization problems can be challenging due to the diverse algorithmic and measurement interfaces. Additionally, as novel algorithms are developed, a standard set of community-implemented algorithms should be used as a benchmark to which algorithmic performance should be compared to. Finally, as algorithms become more complex, it becomes necessary to make these algorithms easily accessible to non-experts such that they can be used in a wide variety of applications. To facilitate the robust development, implementation, and delivery of complex optimization algorithms to accelerator operations teams, the machine learning community has contributed to the Xopt Ecosystem, a collection of modular python packages that provide a framework for developing and deploying advanced optimization algorithms towards solving arbitrary problems.

These packages (shown in Fig. 1) include Xopt [5] which provides a framework for connecting optimization algorithms to arbitrary optimization problems. Xopt also provides a collection of community developed algorithms that can be used off-the-shelf by non-experts or adapted to specific use cases by experts. These algorithms include those historically used by accelerator physicists today for online optimization (Nelder-Mead Simplex, RCDS, Extremum Seeking), algorithms used in massively parallelized contexts such as simulations running on high performance computing clusters (evolutionary algorithms including NSGA-II, CNSGA), and a wide variety of Bayesian optimization based algorithms. The other major package in the Xopt ecosystem is Badger [6], which provides a convenient graphical user interface for Xopt algorithms that can be used in facility control rooms. By separating the implementation of optimization algorithms and the user interface into separate packages, the same optimization algorithms can be used in both online and offline contexts. This modular design, coupled with a flexible framework for interacting with arbitrary problems has contributed to widespread adoption of Xopt across domestic (SLAC, LBNL, LANL, Fermilab, Argonne, BNL, Cornell) and international (STFC Rutherford Appleton Laboratory, ERSF, DESY, Pohang) accelerator facilities.

In this work, we provide an overview of the Xopt ecosystem with an emphasis on usage inside accelerator control rooms for solving online optimization challenges. We then highlight recent use cases at SLAC and other facilities. Finally, we describe future development directions, namely the integration of community developed algorithm specification standards and the addition of advanced algorithms from control theory and machine learning such as reinforcement learning.

* rroussel@slac.stanford.edu

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WESV001

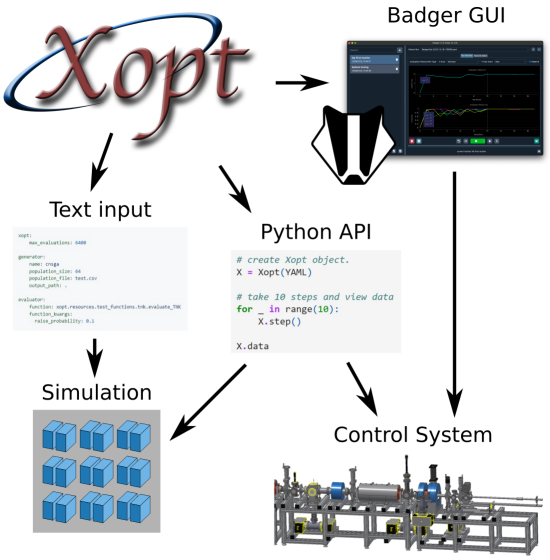

Figure 1: Overview of the Xopt ecosystem highlighting the different methods by which Xopt can be used to solve online and offline optimization problems.

XOPT

Xopt is an open-source Python library that provides a flexible framework for optimization, adaptive experimentation, and online tuning of complex systems. Xopt consists of four main components, the VOCs object, the Generator, the Evaluator, and Xopt object itself. The VOCs object stores the variables, objectives, constraints, and other parameters that define the optimization problem. The generator object implements the optimization algorithm, and when called, produces suggested points in parameter space that should be evaluated next. The evaluator object contains logic to evaluate the objectives and constraints as specified by a user-defined python function. The main Xopt object handles orchestration, namely, controlling when the generator is asked to generate points and controlling how those points are passed to the evaluator. The Xopt object also stores a dataframe of the variable, objective, and constraint values that are collected during optimization. By taking a modular, object-oriented approach Xopt simplifies implementation of algorithms specification and objective function evaluation, while also allowing the individual components to be used independently of one another.

Users connect Xopt to their specific optimization problems by creating a python function that accepts a dictionary of variable values and returns a dictionary that contains objective and constraint values that correspond to a problems' VOCs object. By using this basic, dictionary-based interface, algorithms in Xopt can easily be applied to arbitrary optimization problems. For example, python functions can be used to set the values of power supplies and then measure aspects of the beam distribution on a diagnostic screen. On the other hand, a similar python wrapper such as LUME [7] can be used to evaluate a beam dynamics simulation with a given set of design parameters on a high performance computing cluster. Additionally, by using a dictionary format as inputs and outputs to this python function, users can easily provide or return metadata associated with each measurement such that it is stored in the main Xopt dataframe. In applications at SLAC this metadata can include for example, the measurement time to cross reference with the SLAC archive utility, the location of dump files that contain diagnostic images collected during the course of a emittance measurement, or external beamline parameters not involved in the optimization process for future ML-based model training,

BADGER

Badger is a graphical user interface (GUI) designed to provide a user-friendly front end for optimization workflows, particularly in experimental settings such as accelerator tuning. While Xopt provides the underlying optimization engine—defining problems, executing algorithms, and managing results—Badger makes these capabilities accessible through an interactive, visual interface. Together, they form a complementary pair: Xopt as the flexible backend library and Badger as the GUI layer that lowers the barrier to entry for non-expert users.

The Badger interface allows experimenters and operators in the control room to configure optimization runs without writing code directly. Users can define variables, objectives, and constraints through simple forms and panels rather than Python scripts. Once configured, Badger communicates with Xopt to launch the optimization process, monitor progress in real time, and visualize results as they are collected. This includes the ability to display optimization histories, trade-off curves in multi-objective problems, and convergence behavior across iterations.

Users of badger create their own python classes to connect badger communication with their respective control systems. Badger requires the definition of an "Environment" class, which implements separate methods to get the current value of variables, to set variable values, and to conduct measurements, all of which can be used as objectives and/or constraints. Additionally, lower level "Interface" classes can be defined for common control system protocols, such as EPICS, DOOCS, or Tango, and can be reused in multiple badger environments. By using this common framework for communication between badger and the facility, the GUI can be easily adapted to arbitrary control systems.

EXAMPLE USE CASES

Xopt and Badger have been applied to solving a wide range of optimization problems in both experimental and simulated contexts. Here we provide a few case studies at SLAC that demonstrate their effectiveness.

At SLAC Xopt and Badger have been extensively used to tune the LCLS and LCLS-II beamlines to decrease the beam emittance and increase the free electron laser pulse intensity. Recently, Badger was used in the control room to adjust up to 8 injector parameters (on both the LCLS and LCLS-II beamlines) in order to reduce the beam emittance

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WESV001

while keeping the beam matched to design twiss values at a downstream diagnostic screen. In this case an autonomous emittance measurement scheme, implemented using Xopt, was used to continuously adapt the quadrupole scan sample points such that measurements of the beam size were within a region of interest while still ensuring that the transverse phase advance was sufficient to properly measure the emittance. For both the LCLS and LCLS-II photoinjectors, badger was able to reduce the transverse emittance of both planes by a factor of 2 within the span of 20-30 minutes.

For tuning FEL pulse intensity, the SLAC Badger environment grabs beam synchronous data from the gas detector diagnostics at LCLS and LCLS-II. The gas detector signal is acquired over a period of time as defined by an operator through the badger GUI and the 80th percentile of the measured distribution is computed. This signal is then optimized using Badger with respect to some or all of the matching quadrupoles in the beamline (up to $\sim$ 20 parameters), while constraining the optimization on a maximum level of beam losses. To assist in configuring the optimization routine, Badger can automatically compute percentage deviations from current machine settings such that optimization occurs using small perturbations to machine settings. Using Badger, we have routinely improved the FEL pulse intensity from a cold start (near zero intensity) and from operator tuned setpoints during normal operations as shown in Fig. 2.

Figure 2: Badger graphical user interface display after conducting optimization of the LCLS-II FEL pulse intensity.

From these developments, Badger-based tuning of accelerator parameters was integrated into the restart of both the LCLS and LCLS-II beamlines after the summer downtime this year. The optimization configurations developed during this time were saved using Badger's templates feature such that they can be easily re-run during future restart operations.

FUTURE DEVELOPMENTS

Xopt and Badger will continue to be developed with additional capabilities, improvements to user interfaces / APIs, and enhancements of algorithmic implementations. Slated for early 2026, Xopt v3.0 will be released, with enhancements to the specification of VOCs (discrete parameters, virtual objectives, etc.) and Generators such that they adhere to standards set forth by the Scidac CAMPA Consourtium in order to facilitate interchange of algorithms between Xopt, libensemble [8], and optimus [9] optimization packages. Additionally, the Badger GUI will be further developed to introduce new capabilities, such as loading previous data into the optimization algorithm or making it easier and faster to customize algorithms for specific use cases. Finally, we aim to continue working with the community to add additional algorithms to Xopt including constrained Extremum Seeking [10] and reinforcement learning. These developments are informed by recommendations from the Xopt Steering Committee, an international group of scientists and facility operators interested in ML-based control of accelerators.

CONCLUSION

In this work, we summarized the Python packages that form the Xopt Ecosystem for online and offline optimization of accelerator physics problems. These packages have achieved a level of maturity such that they are used to solve a wide range of problems in accelerator physics and have been included in routine operations at SLAC and elsewhere. While the development of these packages has been motivated by problems in accelerator physics, they can be used off-the-shelf for arbitrary scientific facilities such as telescopes or photon science beamlines.

ACKNOWLEDGMENTS

This work is supported by the U.S. Department of Energy, Office of Science under Contract No. DE-AC02-76SF005152.

REFERENCES

[1] J. A. Nelder and R. Mead, "A simplex method for function minimization", *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965. doi:10.1093/comjnl/7.4.308

[2] X. Huang, "Robust simplex algorithm for online optimization", *Phys. Rev. Accel. Beams*, vol. 21, no.10, p.104601, 2018. doi:10.1103/PhysRevAccelBeams.21.104601

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii", *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. doi:10.1109/4235.996017

[4] R. Roussel *et al.*, "Bayesian optimization algorithms for accelerator physics", *Phys. Rev. Accel. Beams*, vol. 27, no. 8, p. 084801, 2024. doi:10.1103/PhysRevAccelBeams.27.084801

[5] R. Roussel, A. Edelen, A. Bartnik, and C. Mayes, "Xopt: A simplified framework for optimization of accelerator problems using advanced algorithms", in *Proc. IPAC'23*, Venice, Italy, pp. 4847–4850, 2023. doi:doi:10.18429/jacow-ipac2023-thpl164

[6] Z. Zhang *et al.*, "Badger: The Missing Optimizer in ACR", in *Proc. IPAC'22*, Bangkok, Thailand, pp. 999–1002, Jun. 2022. doi:10.18429/JACoW-IPAC2022-TUPOST058

[7] C. Mayes *et al.*, "Lightsource Unified Modeling Environment (LUME), a Start-to-End Simulation Ecosystem", in *Proc. IPAC'21*, Campinas, SP, Brazil, pp. 4212–4215, May, 2021. doi:10.18429/JACoW-IPAC2021-THPAB217

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WESV001

[8] 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 Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 977–988, 2022. doi:10.1109/tpds.2021.3082815

[9] A. Ferran Pousa *et al.*, "Bayesian optimization of laser-plasma accelerators assisted by reduced physical models", *Phys. Rev. Accel. Beams*, vol. 26, no. 8, p. 084 601, 2023. doi:10.1103/PhysRevAccelBeams.26.084601

[10] A. Williams, M. Krstić, and A. Scheinker, "Practically safe extremum seeking", in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 1993–1998, 2022. doi:10.1109/CDC51059.2022.9993069