HYBRID ONLINE OPTIMIZATION FOR HANDS-OFF TUNING OF THE ALS INJECTOR

G. Martino*, T. Hellert, S. C. Leemann, A. Sulc
Lawrence Berkeley National Laboratory, Berkeley, United States

Abstract

Reliable hands-off injector operation calls for fast, sample-efficient tuning under drift and competing goals (e.g., capture efficiency, energy spread, transmitted charge). We present an autotuning framework for the ALS injector combining complementary online optimizers for robust performance under strict machine-protection/operability constraints. The controller alternates methods based on objective structure and information gain, fusing diagnostics across longitudinal and transverse systems. The framework itself is generic: optimization procedures layer on top, and an operator or expert selects the appropriate one for the task—including a cold-start bring-up that recovers beam through staged charge gates, and a close-bounds optimization that refines around a known working point with narrowed search bounds. Both enforce safety via bounded steps and surrogate constraints. Initial studies on the ALS injector show that the automated procedures reach comparable working points faster than operator-driven tuning and with better run-to-run repeatability, while preserving capture within the booster's tight longitudinal acceptance.

INTRODUCTION

The Advanced Light Source (ALS) [1] at Lawrence Berkeley National Laboratory is a DOE user facility, and uninterrupted user delivery during topoff operation requires the injector to feed the booster reliably. Maintaining high performance day to day requires regular tuning across the injector's RF, focusing, steering, and timing systems. Today, this tuning is performed primarily by operators: the process is time-consuming, its results can vary from shift to shift, and it draws on the experience of a small group of experts.

Several method families have proven effective for online accelerator tuning, including Bayesian [2, 3], model-free descent (RCDS) [4], and extremum seeking [5]. On the injector's drifting landscape, where capture efficiency, energy spread, and transmitted charge compete, no single method dominates, and free-running any of them under strict machine-protection bounds wastes time on infeasible proposals. Our goal is hands-off, sample-efficient tuning that is robust under drift and safe by construction. We pursue this through hybrid orchestration of these complementary optimizers, diagnostic fusion across longitudinal and transverse subsystems, and a layered safety envelope that constrains every proposal regardless of which optimizer produced it.

* gmartino@lbl.gov

HYBRID OPTIMIZATION FRAMEWORK

Architecture

The framework, `tuning_scripts` shown in Fig. 1, is a modular monorepo built around a FastAPI [6] + CLI core with a Dash [7] web UI and an EPICS hardware plugin. A plugin layer isolates hardware specifics (live ALS, ALS read-only, and simulator), keeping the optimization core environment-agnostic and portable to other facilities. Redis [8]-backed job tracking lets the web UI, scripts, and agents drive runs over a shared Representational State Transfer (REST) API; the API in turn communicates with Redis via the Redis Serialization Protocol (RESP). The framework is generic by design: predefined optimization procedures—cold-start bring-up, close-bounds refinement, and others—layer on top of the same core, with the operator or expert selecting the appropriate procedure for the task at hand.

Optimizers and Surrogates

The optimizer pool is built primarily on Xopt [9] and spans several method families. Local deterministic descent uses RCDS [4] and Nelder-Mead. Model-based search relies on Bayesian optimization with upper-confidence-bound (UCB) and expected-improvement (EI) acquisitions, complemented by BAxUS [10] for high-dimensional problems on adaptive nested subspaces. Multi-objective and evolutionary search are covered by NSGA-II [11] and its constrained variant CNSGA, with hypervolume-improvement acquisitions [12] for the Bayesian counterpart. Continuous-control tuning uses extremum seeking (ES) [5], which is robust on noisy landscapes. Random sampling and Latin-hypercube sampling (LHS) supply broad exploration. Surrogate models are configurable: a BoTorch [13] Gaussian process is the default for sample-efficient continuous tuning, while tree-based ensembles—random forests [14] and extra trees [15]—scale to larger evaluation budgets and natively handle mixed continuous, categorical, and conditional parameter spaces.

Safety and Operability

Safety is enforced as a layered envelope that is optimizer-agnostic: the same bounds and step-size constraints apply to every optimizer in the pool, so BO, RCDS, ES, and LHS all share the same operability guarantees. Every proposal is constrained to bounded knob ranges and a bounded step size ($\|\Delta x\| \leq \delta$), an approach with prior precedent for safe Bayesian optimization on production accelerators [16]. These constraints are enforced server-side: the API validates each request against the active bounds before actuation,

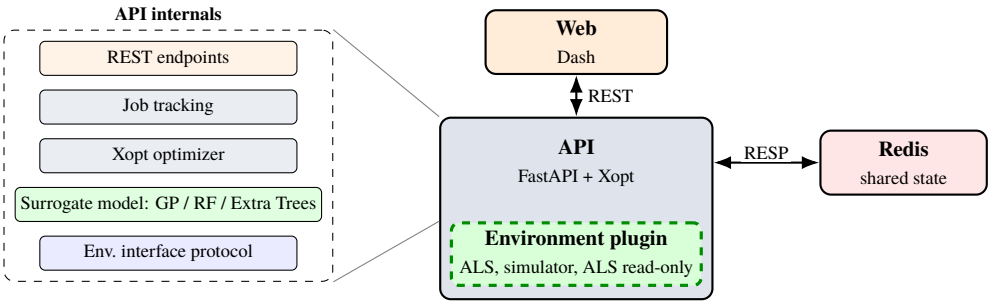


Figure 1: `tuning_scripts` container architecture. The FastAPI core, expanded on the left, runs the optimizer pool, surrogates, and an environment-interface protocol; the environment plugin slot in the lower half loads a hardware backend into the same API process. Clients (Dash web UI, scripts, agents) drive runs over REST, with Redis-backed shared state.

with no client-side bypass available to scripts or agents, and bounds can be narrowed per run for sensitive scenarios. If a bound check fails, the proposal is rejected and the optimizer retries from the current setpoint. Recoverability is built into the orchestration: the initial machine state is saved at the start of every run, so it can always be restored if a run fails or is aborted.

HYBRID OPTIMIZATION ALGORITHM

Two procedures are currently defined: a cold-start bring-up that walks the injector from a zeroed state to beam delivered into the booster, and a close-bounds optimization that refines around a known working point. Both share a common stage structure: each stage seeds the active knob set with Latin-hypercube samples, then refines with a primary optimizer initialized from those samples, and is gated by a downstream charge monitor that must register beam before the procedure advances. The cold-start bring-up runs one or more stages depending on the variant; the close-bounds optimization is a single stage localized to one subsystem, with bounds shrunk around the current best setpoint.

By default, each stage optimizes the charge reading at its gate. We have additionally tested fused objectives that combine the gate charge monitors with BPM signals (X, Y, sum) along the injector transport lines—for example, a linac transmission efficiency (post-linac to pre-linac charge ratio) and a centered-charge metric that weights charge by orbit deviation from a golden orbit—to bias the optimizer away from off-axis solutions that maximize raw charge alone.

Cold-start Bring-up Procedure

Both variants leverage the ALS's archived operational data, taking each variable's search range from its full archived envelope to keep the optimizer within historically operated bounds.

The phase-only variant runs as a single stage tuning ~5 RF-phase knobs against the booster-entry charge monitor, on the assumption that the rest of the machine is tuned; it runs end-to-end without expert intervention, using Bayesian optimization with EI acquisition as the primary optimizer.

The multi-stage variant addresses a fully zeroed machine in three sequential stages, each tuning a progressively wider knob set against a downstream charge monitor. The first stage tunes the gun trigger timing alone to produce charge at the gun output. The second stage jointly tunes the linac-RF and the steering of the linac-to-booster line, with a current transformer in that line as the objective. The third stage opens the search to 16 knobs spanning RF and steering across the injector, driven by the booster-entry charge monitor.

Close-bounds Optimization Procedure

The close-bounds procedure refines around a known working point by narrowing each variable's search bounds to a configurable window centered on the current setpoint, restricting the optimizer to a small neighborhood of the live machine state. The window size is left to the user; in our deployment a width of 10% of each variable's safe range has worked well for routine tuning. The surrogate can be started fresh per run or initialized from a previously learned model, with the choice depending on how representative recent runs are of the current state. Typical use cases are drift compensation and per-shift trim-up.

RESULTS AT THE ALS

The results below come from runs on the live ALS injector, launched from the operator UI and actuated through the EPICS plugin: the phase-only cold-start variant and close-bounds optimization.

Cold-start Bring-up Results

The phase-only variant reliably converges to a high-charge working point. Figure 2 shows a representative run: the Latin-hypercube seeding step distributes samples broadly across all three RF-phase axes, exploring the full safe range for each variable. Bayesian optimization with EI acquisition takes over thereafter, with consecutive proposals tied together by the bounded-step constraint and gradually drawn toward the high-charge basin visible as the yellow cluster of points. Samples settle in a tight neighborhood of the optimum within a few tens of iterations.

Close-bounds Optimization Results

Close-bounds optimization delivers consistent improvements in transmission across the injector chain on top of

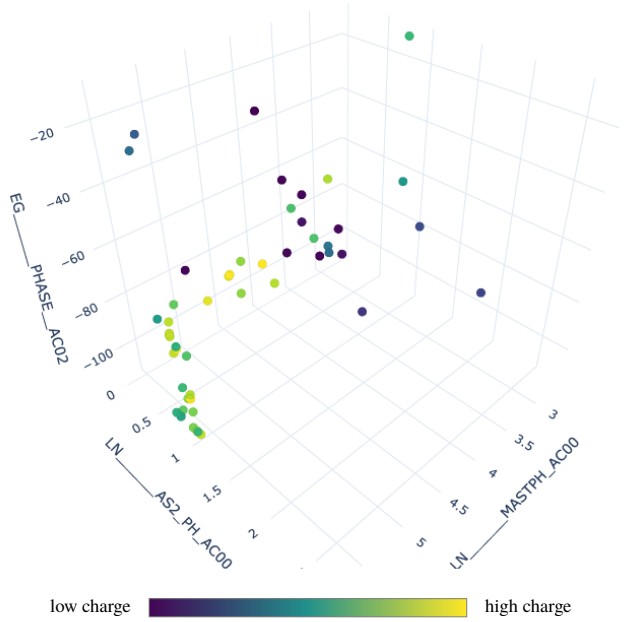
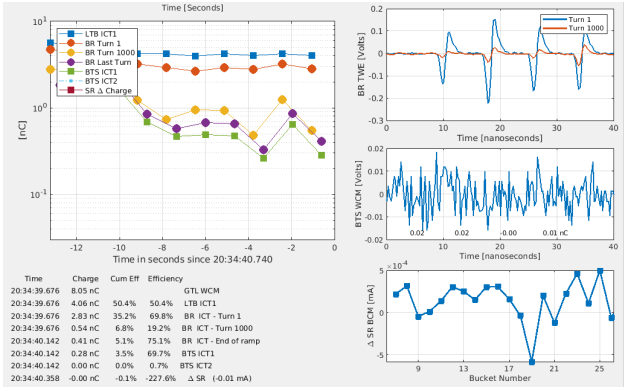


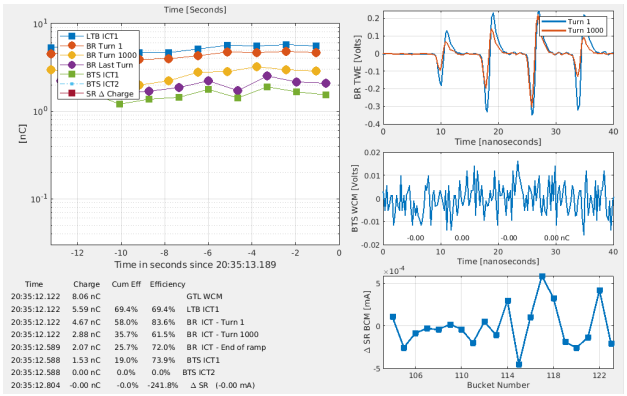
Figure 2: Phase-only bring-up trajectory in a 3-knob slice of the RF-phase parameter space; the gradient bar above maps color to the booster-entry charge objective.

an established working point. To characterize its recovery behavior in a controlled experiment, we set the injector to an off-nominal, low-charge working point and then ran close-bounds optimization from there.

Figure 3 shows the ALS's injection charge monitor [17] before and after the run. Each dot is a single injection shot, and each line traces one of the integrating current transformers (ICTs) along the chain: one in the linac-to-booster (LTB) transport, three inside the booster (sampled at turn 1, turn 1000, and end-of-ramp), two in the booster-to-storage (BTS) transport (ICT1 and ICT2), and one at the injection in the storage ring (SR). After optimization, every monitor reads a higher charge, and the traces stay close together rather than diverging downstream—less charge is lost between successive monitors, indicating better capture and transmission. Charge at BTS ICT1, immediately downstream of the booster, increases from 0.28 nC to 1.53 nC—a factor of ~5.5—while every proposal stays inside the narrowed search bounds.



(a) Before optimization



(b) After optimization

Figure 3: ALS's injection charge monitor before (a) and after (b) a representative close-bounds run. Each dot is a single injection shot; each line traces one monitor along the accelerator chain.

CONCLUSION AND OUTLOOK

We have presented a hybrid online optimization framework for hands-off tuning of the ALS injector, structured around three complementary ideas: orchestrating a pool of online optimizers rather than committing to a single method, fusing diagnostics across the longitudinal and transverse subsystems into objectives that resist off-axis local optima, and enforcing a layered safety envelope that constrains every proposal regardless of which optimizer produced it. Initial deployment on the live ALS injector validates both procedures: the phase-only bring-up converges to a high-charge working point within a few tens of iterations from a Latin-hypercube seed, and close-bounds optimization recovers a factor of ~5.5 in charge at the booster exit from an off-nominal starting working point while every proposal stays inside its narrowed bounds.

The current procedures treat each tuning run as an independent optimization. In production, the injector working point drifts between shifts and across thermal cycles, and we are pursuing extensions that target this non-stationarity directly: tracking a moving Pareto front during drift, and time-decayed sample weighting to limit surrogate staleness as historical observations age relative to the current machine state. `tuning_scripts` is open-source and machine-agnostic by construction; the plugin contract that decouples the live machine from the simulator is intended to lower the cost of porting these procedures to other facilities and control systems.

ACKNOWLEDGEMENT

This work was supported by the Director of the Office of Science of the U.S. Department of Energy under Contract No. DEAC02-05CH11231.

REFERENCES

[1] T. Hellert *et al.*, "Status of the Advanced Light Source", in *Proc. 15th Int. Part. Accel. Conf. (IPAC'24)*, Nashville, TN, USA, May 2024, pp. 1309–1312. doi:10.18429/JACoW-IPAC2024-TUPG37

[2] J. Duris *et al.*, "Bayesian optimization of a free-electron laser", *Phys. Rev. Lett.*, vol. 124, p. 124801, 2020. doi:10.1103/PhysRevLett.124.124801

[3] R. Roussel, A. Hanuka, and A. Edelen, "Multiobjective Bayesian optimization for online accelerator tuning", *Phys. Rev. Accel. Beams*, vol. 24, p. 062801, 2021. doi:10.1103/PhysRevAccelBeams.24.062801

[4] X. Huang, J. Corbett, J. Safranek, and J. Wu, "An algorithm for online optimization of accelerators", *Nucl. Instrum. Methods Phys. Res. A*, vol. 726, pp. 77–83, 2013. doi:10.1016/j.nima.2013.05.046

[5] A. Scheinker, X. Pang, and L. Rybarcyk, "Model-independent particle accelerator tuning", *Phys. Rev. ST Accel. Beams*, vol. 16, p. 102803, 2013. doi:10.1103/PhysRevSTAB.16.102803

[6] S. Ramírez, "FastAPI: a modern, high-performance web framework for building APIs with Python type hints", Software, Accessed 2026-05, 2024, `https://fastapi.tiangolo.com/`,

[7] Plotly Technologies Inc., "Dash: low-code framework for building analytical web apps in Python", Software, Accessed 2026-05, 2024, `https://dash.plotly.com/`,

[8] S. Sanfilippo and Redis contributors, "Redis: an in-memory data structure store used as a database, cache, and message broker", Software, Accessed 2026-05, 2024, `https://redis.io/`,

[9] R. Roussel, C. Mayes, A. Edelen, and A. Bartnik, "Xopt: a simplified framework for optimization of accelerator problems using advanced algorithms", in *Proc. 14th Int. Part. Accel. Conf. (IPAC'23)*, Venice, Italy, May 2023, pp. 4847–4850. doi:10.18429/JACoW-IPAC2023-THPL164

[10] L. Papenmeier, L. Nardi, and M. Poloczek, "Increasing the scope as you learn: adaptive Bayesian optimization in nested subspaces", in *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 11586–11601, 2022.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002. doi:10.1109/4235.996017

[12] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization", in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pp. 9851–9864, 2020.

[13] M. Balandat *et al.*, "Botorch: A framework for efficient monte-carlo Bayesian optimization", in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pp. 21524–21538, 2020.

[14] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration", in *Learning and Intelligent Optimization (LION 5)*, vol. 6683, pp. 507–523, 2011. doi:10.1007/978-3-642-25566-3_40

[15] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees", *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006. doi:10.1007/s10994-006-6226-1

[16] J. Kirschner, M. Mutný, A. Krause, J. Coello de Portugal, N. Hiller, and J. Snuverink, "Tuning particle accelerators with safety constraints using Bayesian optimization", *Phys. Rev. Accel. Beams*, vol. 25, p. 062802, 2022. doi:10.1103/PhysRevAccelBeams.25.062802

[17] G. Portmann, J. Corbett, and A. Terebilo, "An accelerator control middle layer using MATLAB", in *Proc. 2005 Particle Accelerator Conf. (PAC'05)*, pp. 4009–4011, 2005. doi:10.1109/PAC.2005.1591699