ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP030

INTEGRATION OF AN OPTIMIZER FRAMEWORK INTO THE CONTROL SYSTEM AT KARA

C. Xu*, E. Blomley, A. Santamaria Garcia, and A.-S. Müller
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
M. Zhang
Princeton University, Princeton, United States

Abstract

Tuning particle accelerators is not straightforward as they depend on a large number of non-linearly correlated parameters that drift over time. In recent years advanced numerical optimization tools have been developed to assist human operators in tuning tasks. A proper interface between the optimizers and the control system will encourage their daily use by the accelerator operators. In this contribution, we present our latest progress in integrating an optimizer framework into the control system of the KARA storage ring at KIT, which will allow the automatic tuning methods to be applied for routine tasks.

INTRODUCTION

The Institute of Beam Physics and Technology (IBPT) at Karlsruhe Institute of Technology (KIT) operates several accelerators, including the Karlsruhe research accelerator (KARA) and the far-infrared linac and test experiment (FLUTE) [1], along with a compact storage ring cSTART and a plasma accelerator that are in the planning phase. Various machine learning methods have been implemented and studied at the accelerators of IBPT. For example, Bayesian optimization was used to optimize the injection efficiency into the storage ring [2], and reinforcement learning (RL) methods were employed to control the transverse motion and microbunching instability at KARA [3, 4]. In order to apply those methods at the accelerators, one often needs to implement specialized wrappers, such as the interface to the control system and results logging. They often don't have a user interface, which makes it more difficult for operators to use. In addition, the lack of common interfaces makes the software harder to maintain and requires the developer to be up-to-date with control system or operating system updates. Therefore, it is desired to have a generalizable framework for accelerator tuning tasks that can make common features and functionalities available across different particle accelerators.

One of the early examples of such a framework is OCELOT [5, 6], a multi-purpose software suite developed for the operation of European X-ray free-electron laser (EuXFEL). It contains a generic optimization framework that implemented several ready-to-use tuning algorithms like Nelder-Mead simplex, Bayesian optimization (BO) [7], and extremum seeking (ES) [8]. Nevertheless, many design choices of OCELOT have been made around FEL operation and are more difficult to adapt to other tasks.

Recently, Badger [9] was developed as an alternative generic optimizer framework that focuses on efficiency and user-friendliness in the accelerator control room. After being tested at several facilities it has shown promising results in accelerator tuning tasks. At IBPT, we are continuously working on increasing the level of automation in daily operations and incorporating more standardized packages [10]. We decided to use Badger as the optimization framework for its simplicity and adaptability to new tasks. In this paper, we share our experience adapting Badger to the control system at KARA and demonstrate the first applications of automatic tuning during the accelerator commissioning process.

INTEGRATION OF BADGER

Badger Components

Here we briefly describe the the architecture of Badger and the steps needed to apply Badger to a new optimization task. Badger is written in Python with a modular approach, where the core package merely provides the basic functionality like a command line interface (CLI) and a graphical user interface (GUI) to control, monitor, and log the optimization process. It manages interfaces, environments, and algorithms through a plugin system, which allows easy modification and development of new plugins.

Interface In Badger, the communication with the control system is defined in an interface. Standard interfaces like *pyepics* and *pydoocs* are already implemented in Badger, providing basic read and write functionalities for the process variables (PVs). For the applications at KARA, we use an in-house developed Python package [11] for the channel access (CA) to our EPICS control system, which takes site-specific structures into account and uses the *caproto* [12] package as a backbone.

Environment The environment defines all the available variables and observables of interest for a subsystem of the accelerator. In the case of the EPICS control system, it contains mostly a list of the PVs that will be used in an optimization process.

Algorithm The algorithm is used to suggest the next point to evaluate in the optimization process. Since the interfaces with the machine are sufficiently abstracted, numerical optimization methods can be separately developed and easily incorporated into Badger. For example, the Xopt [13]

* chenran.xu@kit.edu

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP030

package can be seamlessly loaded in Badger. It provides popular tuning methods such as ES, Nelder-Mead simplex, and a wide range of Bayesian optimization variants, including adaptive BO, trust-region Bayesian optimization (TurBO), and multi-objective BO.

Routine The routine in Badger defines a specific optimization task. It contains the environment and the optimization algorithm to use. From the environment, it selects which of the parameters are considered actuators, objectives, and constraints for the task. This can be loaded through a YAML file or dynamically configured using the CLI/GUI during the operation.

Containerized Deployment

We chose to deploy Badger as a containerized application for the sake of maintainability and stability. In such a way, it is not affected by future changes in the operating systems of our control room consoles. The workflow to deploy the Badger container is illustrated in Fig. 1.

We maintain two public copies of the Badger project, the core package *Badger* for custom modifications and the *badger-plugin*, where we implemented additional environments and interfaces for the KARA tuning tasks. Both repositories are forked from the original GitHub projects and synchronized to the KIT Gitlab instance via Pull&Push mirrors.

Another repository *badger-containerized* contains the files to set up a docker container. A continuous integration and continuous development (CI/CD) pipeline is configured so that once the code is updated, a runner will automatically build the docker image and push it to our institute's container registry. In the docker image, it clones our latest custom version of Badger and its plugins in Gitlab and installs the dependencies from our facility-specific package registries, such as the custom caproto and Elog packages [11].

Apptainer[1] [14] is used as the container system in the control room consoles and managed centrally via the Salt system [15, 16]. It can naturally execute Docker images and has been widely adopted as a container format in the scientific community. The operators can start the Badger container in the control room directly using Apptainer, which will pull the latest image version from the container registry, or use the local copy otherwise.

When starting the container, we mount a directory in the console into the container to store the configured routines and the results of the optimization runs produced by Badger. This folder can be shared among the control room consoles so that one can access the Badger package from any device. In the future, we plan to set up a workflow so that the optimization runs are automatically stored in a database.

Automatic Posting in Electronic Logbook

We can adapt the package further to our needs and add custom features as we maintain a copy of Badger in our Gitlab repository. One of the first customizations we implemented was the logbook function. At IBPT, we have an electronic logbook system called ELog [17]. In addition to the manual logging, automatic logbook entries are generated by standard measurement routines, such as tune, chromaticity, dispersion, and orbit response matrix measurements.

The Badger package provides its own logbook functionality to save locally a screenshot of the optimization process and an XML file with the run configurations. We adopted this implementation so that, after a successful optimization run, an Elog entry is generated with the attributes filled with the respective settings read from the Badger routine. For the automatic logging, we configured a logbook with the following attributes.

- *Fill number*: An integer indicating the run number of beam operations at KARA, incremented every time the storage ring is injected with the new beam.
- *Environment*: Name of the Badger *environment*.
- *Actuators*: List of the PVs used for the optimization and their ranges respectively.
- *Objective*: PVs or the metric used as the objective function for optimization.
- *Optimizer*: Name of the algorithm used for optimization, as defined in Badger and Xopt generators.
- *Start time*: Unix timestamp of the optimization start time.
- *End time*: Unix timestamp of the optimization end time.
- *Attachment*: The screenshot of the optimization process, and the routine configuration in an XML file, as defined in the original Badger package.

While we already have a Cassandra archiving system [18] to store the history of machine parameters, the Elog provides metadata to the optimization runs, such as the environment configuration and algorithm hyperparameters. An electronic logbook with these attributes will facilitate easy filtering and finding of desired optimization runs for future analysis.

ACCELERATOR TUNING DEMONSTRATION

Simulation Model

The Badger functionalities are first tested in a simulated environment. We used the orbit correction system at KARA [19], which includes a SoftIOC [20] to simulate the orbit responses with respect to the corrector magnets settings. In the corresponding Badger environment, we included the corrector settings as variables and beam position monitor (BPM) readings as observations. Different metrics can be calculated based on the BPM values, e.g. the mean square error (MSE) and the mean absolute error (MAE) with respect to the target orbit.

Figure 2 shows one of the optimization results using BO upper confidence bound (UCB). We randomly offset two corrector magnets to create an orbit distortion and tried to use three other correctors to minimize the MSE of the closed orbit. After some exploration steps, BO quickly converged to a stable setting, taking only around 20 steps. The MSE of

[1] Formerly known as Singularity

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP030

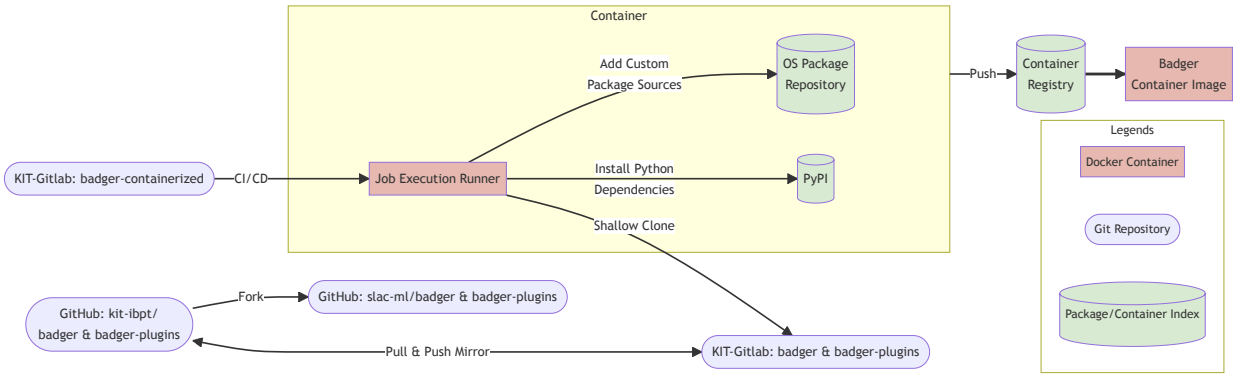

Figure 1: Technical diagram of the containerized Badger deployment workflow. Arrows denote the origin points of the actions.

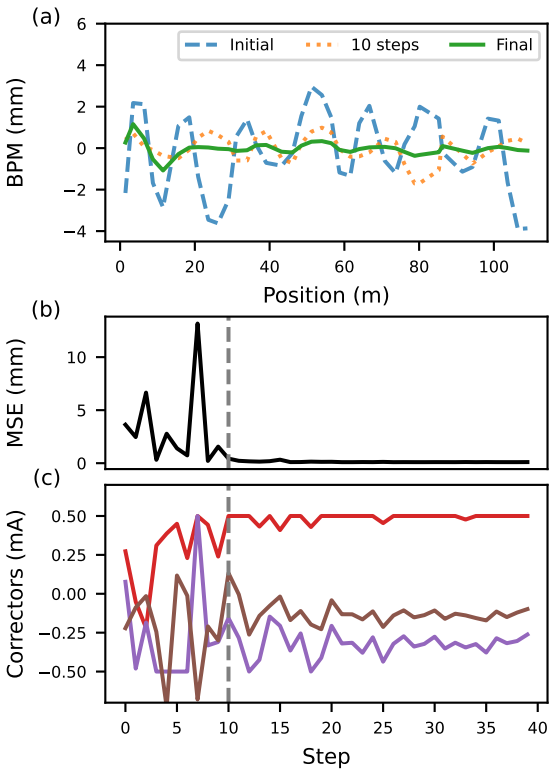

Figure 2: Using Badger to correct the beam orbit in simulation. (a) Beam orbit before, during, and after the optimization process. (b) Progress of the objective mean square error of the BPM readings. (c) Values of the three corrector magnets used to optimize the beam orbit. The gray lines in (b,c) mark the values after 10 optimization steps corresponding to the orange orbit in (a).

the final orbit could be reduced to 0.1 mm, where the initial larger deviations were due to the two offset correctors.

Accelerator Commissioning

During the summer, the KARA accelerator complex was shut down for scheduled maintenance, hardware upgrades, and the exchange of components. After such a shutdown period, a re-commissioning phase is required. A substantial amount of time is often required to manually adjust the parameter settings to start up the machine and optimize the beam injection to the same level of performance as before the shutdown. The injection chain of KARA consists of an electron gun, a racetrack microtron, and a booster synchrotron [21]. The beam passes through the linac in the microtron 10 times and is accelerated up to 53 MeV. Then, it is extracted into the booster synchrotron, further accelerated to 500 MeV, and injected into the storage ring. The injection happens at 1 Hz repetition rate. As this process is inherently sequential, the injection tuning task can be further split into several sub-tasks, which can be optimized after each other. These include

- turn-by-turn beam current optimizations in the racetrack microtron,
- beam transport optimization of the injection line from microtron to booster,
- beam orbit optimization to minimize the beam loss in the booster synchrotron,
- beam transport optimization of the extraction line from booster to the storage ring, and
- injection efficiency optimization into the KARA storage ring.

We designed Badger environments for these tasks and tested several automatic tuning routines in the commissioning stage.

Figure 3 is a screenshot of the Badger run monitor in the GUI, showing one of the microtron optimization runs. The goal is to maximize the signal measured by a current transformer using four corrector magnets as tuning variables. The top panel visualizes the progress of the objective function, in this case, a beam current signal. The values of the tuning variables, which are normalized to their allowed ranges respectively, are plotted in the bottom panel. The middle panel shows the states, which contain signals to be monitored in addition to the objective and variables. The BO algorithm is again used in this case and could converge within a few tens of steps, reaching double the starting current value.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP030

Figure 3: Screenshot of the Badger run monitor during microtron tuning. In the top panel, the objective progress is plotted. Additional observations (states) are monitored in the middle. The bottom panel shows the actuator values.

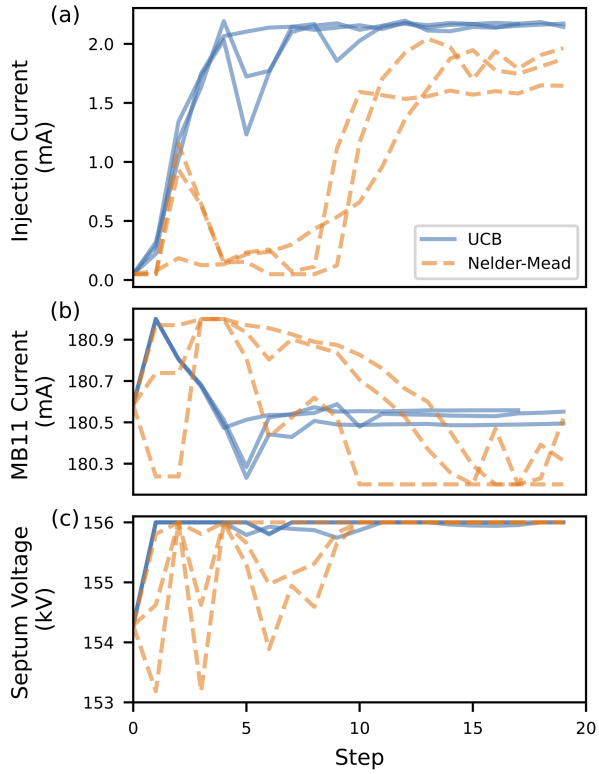

Figure 4: Tuning the booster injection current with Badger. (a) Shows the injection current over steps using the BO-UCB (blue) and Nelder-Mead simplex (orange) methods. (b,c) Show the changes of the actuators respectively, i.e. the extraction bending magnet current and the injection septum magnet voltage.

Nevertheless, the generic optimization approach offered by Badger has also its limitations. We observed that after cycling the magnet in the microtron, the injection performance was much worse than the previously achieved value. This is due to the magnetic hysteresis, which is not accounted for during numerical optimizations. Although it does not affect the beams with higher energies as much, the hysteresis effect becomes prominent for low-energy electrons in the microtron.

We then used Badger to further tune the magnets in the injection line to increase the booster injection current. To test the reproducibility of the results, we repeated one of the usual booster injection tuning tasks using the BO UCB and Nelder-Mead simplex algorithms. Each algorithm was run three times starting from the same detuned initial magnet settings, where no beam was injected into the booster. As shown in Fig. 4, both methods could consistently optimize the injection current within 20 steps, where UCB outperformed Nelder-Mead both in efficiency and the final current value. It needs to be noted that some actuators converged towards the upper limit of the allowed range, which was derived from the operators' experience. In the future, we will fine-tune the optimization range of each component based on the analysis of historic machine settings.

CONCLUSION AND OUTLOOK

We integrated the generic tuning framework Badger and Xopt at the KARA storage ring. A workflow was established to automatically deploy the latest version of Badger in our control system, making use of the CI/CD pipeline, container system, and custom package registries. This can be further used to ease the integration of other external packages into our control system. Multiple Badger environments and routines are implemented for both the simulation and the real accelerator. Initial tests of Badger during the commissioning of the injectors showed promising results. Although it still required human intervention or fine-tuning in some edge cases, such as hysteresis or poorly defined optimization ranges, it still aided the operator and largely automated the injection-tuning process. We expect that in the future, Badger will be used as a routine tuning tool during the daily operation. A well-defined sequence of Badger routines can be designed once we have identified all the necessary tuning steps in the process of starting up the machine. Further algorithms, for example, the hysteresis-aware optimization, will be implemented to deal with more specific tuning cases that are not covered by the ones available in the Xopt package. With these improvements, the storage ring operation can be greatly automated, or even become turn-key.

Furthermore, we plan to make Badger also available for the linac test facility FLUTE and other future accelerators at IBPT. This standardized optimization framework will allow the easy transfer of developed algorithms between different accelerators.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP030

ACKNOWLEDGMENTS

We thank Z. Zhang from SLAC for providing useful suggestions to implement Badger. We thank A. Mochihashi and J. Steinmann for their support during the beam time at KARA, J. Gethmann for the support on the git repositories, and S. Kötter for the support on the orbit correction simulation software. C. Xu acknowledges the support by the DFG-funded Doctoral School "Karlsruhe School of Elementary and Astroparticle Physics: Science and Technology".

CODE AVAILABILITY

The public forks of the code used in this paper are available on GitHub [22, 23]. The code in our Gitlab can be accessed upon request.

REFERENCES

[1] M. J. Nasse *et al.*, "FLUTE: A versatile linac-based THz source", *Rev. Sci. Instrum.*, vol. 84, no. 2, p. 022 705, 2013. doi:10.1063/1.4790431

[2] 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*, vol. 26, no. 3, p. 034 601, 2023. doi:10.1103/PhysRevAccelBeams.26.034601

[3] W. Wang *et al.*, "Accelerated Deep Reinforcement Learning for Fast Feedback of Beam Dynamics at KARA", *IEEE Trans. Nucl. Sci.*, vol. 68, no. 8, pp. 1794–1800, 2021. doi:10.1109/TNS.2021.3084515

[4] L. Scomparin *et al.*, "A Low-Latency Feedback System for Electron Beam Control with Reinforcement Learning", in *Proc. IPAC'23*, Venice, Italy, 2023. doi:10.18429/JACoW-IPAC2023-THPL027

[5] I. Agapov, G. Geloni, S. Tomin, and I. Zagorodnov, "Ocelot: A software framework for synchrotron light source and fel studies", *Nucl. Instrum. Methods Phys. Res. A*, vol. 768, pp. 151–156, 2014. doi:10.1016/j.nima.2014.09.057

[6] S. Tomin, I. Agapov, W. Decking, G. Geloni, M. Scholz, and I. Zagorodnov, "On-line Optimization of European XFEL with OCELOT", in *Proc. ICALEPCS'17*, Barcelona, Spain, 2018, pp. 1038–1042. doi:10.18429/JACoW-ICALEPCS2017-WEAPL07

[7] J. Duris *et al.*, "Bayesian optimization of a free-electron laser", *Phys. Rev. Lett.*, vol. 124, no. 12, p. 124 801, 2020. doi:10.1103/PhysRevLett.124.124801

[8] A. Scheinker *et al.*, "Model-independent tuning for maximizing free electron laser pulse energy", *Phys. Rev. Accel. Beams*, vol. 22, no. 8, p. 082 802, 2019. doi:10.1103/PhysRevAccelBeams.22.082802

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

[10] A. Eichler *et al.*, "First Steps Toward an Autonomous Accelerator, a Common Project Between DESY and KIT", in *Proc. IPAC'21*, Campinas, Brazil, May 2021, pp. 2182–2185. doi:10.18429/JACoW-IPAC2021-TUPAB298

[11] J. Gethmann *et al.*, "Simple Python Interface to Facility-Specific Infrastructure", in *Proc. PCaPAC'22*, Dolní Brežany, Czech Republic, 2023, paper THPP9, pp. 51–53. doi:10.18429/JACoW-PCaPAC2022-THPP9

[12] Caproto, https://github.com/caproto/caproto

[13] R. Roussel, C. Mayes, A. Edelen, and A. Bartnik, "Xopt: A simplified framework for optimization of accelerator problems using advanced algorithms", presented at IPAC'23, Venice, Italy, May 2023, paper THPL164, unpublished.

[14] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute", *PLOS ONE*, vol. 12, no. 5, pp. 1–20, 2017. doi:10.1371/journal.pone.0177459

[15] Salt project, https://saltproject.io/

[16] E. Blomley, J. Gethmann, M. Schuh, A.-S. Müller, and S. Marsching, "Management of EPICS IOCs in a distributed network environment using salt", presented at ICALEPCS 2023, Cape Town, South Africa, 2023, paper THPDP020, this conference.

[17] Elog, https://elog.psi.ch/elog/

[18] S. Marsching, "Scalable Archiving with the Cassandra Archiver for CSS", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, 2013, pp. 554–557. https://accelconf.web.cern.ch/icalepcs2013/papers/tuppc004.pdf

[19] S. Koetter, "Online fit of an analytical orbit response matrix model for orbit correction and optical function derivation", in *Proc. IPAC'23*, Venezia, 2023, pp. 4424–4427. doi:10.18429/jacow-ipac2023-thpl025

[20] Pythonsoftioc, https://github.com/dls-controls/pythonSoftIOC

[21] D. Einfeld, S. Hermle, E. Huttel, R. Rossmanith, and R. Walther, "The injection scheme for the ANKA storage ring", in *6th European Particle Accelerator Conference (EPAC 98)*, 1998, pp. 2135–2137.

[22] KIT-IBPT/Badger, https://github.com/KIT-IBPT/Badger

[23] KIT-IBPT/Badger-plugins, https://github.com/KIT-IBPT/Badger-Plugins