ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M03A001

OPTIMISATION OF THE TOUSCHEK LIFETIME IN SYNCHROTRON LIGHT SOURCES USING BADGER*

S.M. Liuzzo†, N. Carmignani, L. Carver, L. Hoummi,
D. Lacoste, A. Le Meillour, T. Perron, S. White, ESRF, Grenoble, France
I. Agapov, M. Boese, L. Malina, E. Musa, J. Keil, B. Veglia
Deutsches Elektronen-Synchrotron, 22607 Hamburg, Germany
T. Hellert, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
A. Edelen, P. Raimondi, R. Roussel, Z. Zhang, SLAC, Menlo Park, CA, USA

Abstract

Badger [1] is a software designed to easily access several optimizers (simplex, RCDS [2], Bayesian optimization, etc.) to solve a given multidimensional minimization/maximization task. The Badger software is very flexible and easy to adapt to different facilities. In the framework of the EURIZON European project, Badger was used for the EBS and PETRA III storage rings interfacing with the Tango and TINE control system. Among other tests, the optimisations of Touschek lifetime was performed and compared with the results obtained with existing tools during machine dedicated times.

INTRODUCTION

In the framework of the EURIZON European project, a collaboration has been established between DESY and ESRF to work on the optimization of beam parameters for storage ring (SR) light sources such as lifetime and injection efficiency. In a first approach, the Extremum Seeking algorithm was tested in both SRs. It was soon found to be difficult to tune and did not bring major improvements for operation [3]. Contrary to this initial attempt, the software Badger [1] developed at SLAC has demonstrated to be extremely easy to set up, flexible to use and user friendly. It provides access to the Xopt algorithm library [4]. We present in these pages the user experience (UX) and the experimental results achieved thanks to the use of Badger and Xopt at the ESRF-EBS and DESY-PETRA III storage rings.

XOPT

Xopt [4] is a high level python [5] package developed at SLAC National Accelerator Laboratory that provides a simple to use framework for connecting black box optimization algorithms with arbitrary optimization problems. Xopt decomposes optimization problems into three key components: defining parameter spaces and objectives, specifying how to evaluate these objectives, and implementing optimization algorithms. The `VOCS` class defines the optimization space, including the variables, objectives, constraints and constants (statics). The `Evaluator` class defines how to evaluate objectives and constraints given points passed to it using serial or parallel (multithreading, MPI etc.) processes. Finally, the `Generator` class implements the optimization algorithm, and is used to generate points in variable space to be evaluated. The main `Xopt` object choreographs the execution and communication between these three modules in order to perform an optimization cycle with the `step()` command.

This modular, object-oriented approach enables easy modification and customization of optimization routines for specific use cases. For example, the same optimization algorithm can be applied to both simulation and experiment, or shared between different accelerator facilities by swapping out the `Evaluator` object. On the other hand, generators can also be swapped out to compare the performance of different algorithms on the same optimization problem. These objects can also be sub-classed to customize evaluation or generation of points to solve specific problems.

Optimization algorithms are defined by `Generator` objects, which are used to generate future points to be evaluated by calling their `generate()` method. While users are free to implement their own optimization algorithms, `Xopt` comes pre-packaged with a number of conventional and advanced optimization algorithms tuned by experts to be applicable to a wide variety of optimization problems "off-the-shelf". Currently these algorithms include

- Autonomous Characterization
  - Bayesian Exploration [6]
- Single Objective Optimization
  - Nelder-Mead Simplex [7]
  - Robust Conjugate Direction Search [8]
  - Extremum Seeking [3]
  - Upper Confidence Bound BO [9]
  - Expected Improvement BO [10]
  - Trust Region BO (TuRBO) [11]
  - Multi-fidelity BO [12]
- Multi-Objective Optimization
  - Continuous NSGA-II [13]
  - Expected Hypervolume Improvement BO [14]

* This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 871072
† simone.liuzzo@esrf.fr

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO3A001

– Multi-Generation Expected Hypervolume Improvement BO [15]
– Muli-Fidelity Expected Hypervolume Improvement [12]

These generators can be used or modified via Python code inside or outside of `Xopt`. The `Xopt` framework has already been used at a wide number of accelerator facilities and institutions including LCLS, LCLS-II, FACET-II, Cornell University, University of Chicago, LBNL, AWA and DESY.

BADGER

The Badger [16] package enables the utilization of `Xopt` for real-time accelerator tuning by offering both graphical user interface (GUI) and command-line interface (CLI) for experiment operators. A view of the Badger *run monitor* panel during a single objective optimization is presented in Fig. 1.

Figure 1: Badger application run monitor panel. The objective of minimization is displayed in the top plot, while the relative change of the parameters used for tuning is presented in the bottom plot. Users can easily customize the plotting preferences at any time through the toolbar, including options like using time as the x-axis instead of the iteration index or displaying normalized values of the variables.

The architecture of the latest version of Badger is depicted in Fig. 2. Badger establishes connections between the generators (optimization algorithms) in `Xopt` and the underlying machine through its custom *interface* (which handles low-level and fundamental communication with the control system) and *environment* (containing optimization-related information about the machine, such as variables, tuning ranges, and observables). It provides a user-friendly GUI that allows users to easily fine-tune optimization properties, including selecting tuning knobs, configuring tuning ranges, and defining objectives and constraints.

Users can monitor the optimization progress through Badger's *run monitor*, available in both GUI and CLI modes. Furthermore, run data is automatically archived in databases after each iteration of the optimization to ensure no data loss.

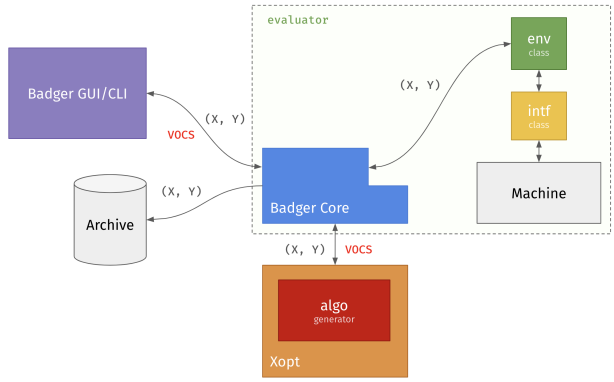

Figure 2: Badger architecture. Badger wraps the machine through an interface and environment, forming an evaluator. It collects inputs from users through the GUI and transforms them into VOCS and algorithm configurations, then utilizes Xopt to drive the optimization process. Badger monitors the data flow, archives data after each step, and provides users with real-time optimization progress during the run.

Badger was specifically designed to meet the needs of the accelerator control room (ACR). In the ACR, it is common to repeat the same tuning procedure multiple times to achieve satisfactory results or to perform it daily to maintain the machine's state at an optimal level. Badger addresses this demand by abstracting the tuning procedure into a *routine*. Users define a routine by selecting the algorithm and the environment, and then configure the `VOCS` as well as the parameters of the algorithm and environment. Once a routine is created, it can be executed at any time with a single button press in Badger. Users can also review the historical runs corresponding to that routine through the Badger *run history browser* for post-run analysis.

To use Badger, the user is required to define two key components:

1. An interface to the specific control system (such as EPICS [17], DOOCS [18], and Tango [19]).

2. An environment that describes the machine to be tuned, including variables, tuning ranges, and observables/measurements.

It's important to note that the environment in Badger serves as an abstraction of the machine rather than a specific problem. For instance, consider the LCLS Badger environment [20], which includes tens of quadrupoles, solenoids, sextupoles, and various other tuning knobs. Additionally, it defines different observables like FEL pulse energy, electron beam size, and electron bunch length. As a result, the same environment can be applied to various optimization scenarios by configuring the `VOCS` accordingly. However, when the environment becomes excessively large, containing thousands of variables, it can be beneficial to divide it into smaller sub-environments (typically optimized individually), thereby enhancing the overall performance and UX of Badger.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M03A001

EXPERIMENTAL RESULTS

Set Up for EBS SR

To set up Badger for the ESRF-EBS storage ring, an *interface* is coded to act on the Tango control system. The interface code is rather simple and it is mostly a copy of the existing Badger examples. For reproducibility, the interface prepared for EBS is using set points (not the read out from the Power Supplies (PS)) to tune magnets. The two functions `get_value` and `set_value` act as a middle layer between Badger and any control system, and make the rest of the code control system agnostic.

In the interface simply replacing two lines of code (`TANGO_HOST`) allows to switch between the accelerator control and the EBS-simulator [21] for test of the newly defined *environments*. Several EBS dedicated environments were created to reproduce the existing ESRF tuning techniques:

- skew quadrupoles for vertical emittance tuning (used only for first test and to debug the code in the EBS-simulator)
- sextupole knobs for lifetime or losses optimization
- octupole knobs for lifetime or losses optimization
- sextupole and octupole knobs for lifetime or losses optimization [22]

Each environment contains information on the devices to tune and limits their values within a maximum range for the optimization. In the EBS storage ring there are 192 sextupoles and 64 octupoles. Tuning each of them independently in an empiric way is not feasible within a reasonable amount of time. Consequently, *knobs* are defined as arrays of sextupole or octupole strengths. For example all focussing sextupoles powered following the amplitude of a cosine function along the SR circumference is a knob. About 200 such knobs were studied before using them for optimizations as detailed in [22]. The most effective knobs were retained based on dynamic aperture and lifetime simulations. A final selection was based on the electron beam response in terms of lifetime and injection efficiency and allowed to limit the optimizations to 24 sextupole knobs and 4 octupole knobs. A few examples out of the 24 knobs of 192 sextupoles used for the EBS SR optimizations are shown in Fig. 3.

A Badger environment was set up to tune the 24 sextupole knobs and 4 octupole knobs simultaneously. This was also possible thanks to the existence in the EBS control system of vectorized arrays of magnet multipoles that synchronously send strengths set point to all magnets. The Badger environment described above includes all the knobs used in usual optimizations at ESRF and thus gives the possibility to compare the existing EBS lifetime tuning technique (run-by-hand, see later) to potentially all algorithms available via Badger and `Xopt` using a common set of variables.

For EBS the target of the optimizations is either total losses reduction all around the SR or lifetime increase. Losses are measured at 128 independent beam loss detectors [23]. Lifetime is obtained either from a current transformer or from beam position monitors, the latest being more

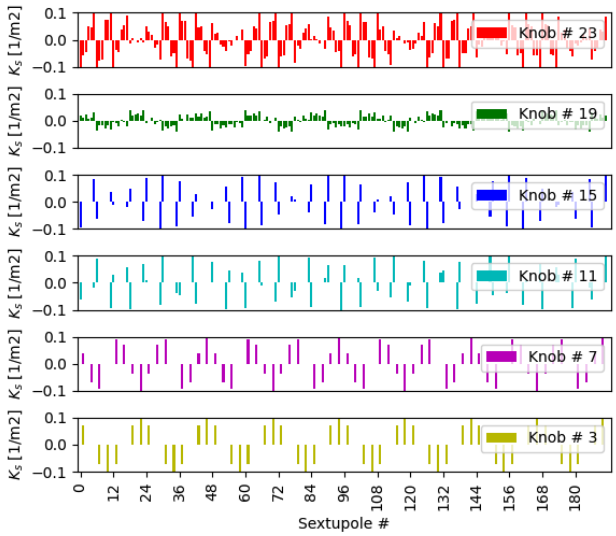

Figure 3: Example of sextupole knobs used for lifetime optimizations of the EBS storage ring.

reactive to changes. During the course of an optimization the beam current naturally decays. The vertical emittance is kept constant with a vertical shaker to 10 pm rad. The orbit is also constantly corrected by the slow (0.1 Hz) and fast (1 kHz) orbit correction simultaneously active. This leaves only the current as a parameter for normalization. Touschek lifetime τ is normalized as

$$\tau^{\text{norm}} = \tau(t)\frac{I(t)}{I_0} \tag{1}$$

and consequently the total beam losses L as:

$$L^{\text{norm}} = L(t)\left(\frac{I_0}{I(t)}\right)^2$$

where I_0 is the current at the optimization startup and assuming that an arbitrary number of bunches in the storage ring is filled with approximately the same current per bunch.

Excluding the three hours necessary to reach full beam polarization after a full current refill (0 to 200 mA, at sample ~250 in Fig. 4), the simple Touschek lifetime normalization proposed above works as expected: the normalized lifetime is almost constant despite the current decay as shown in Fig. 4. In this figure also the hor. and ver. emittance dependence and vacuum lifetime contribution are removed from the measured total lifetime. These two parameters are constant during the optimization process and thus irrelevant.

In practice the optimizations at ESRF are done targeting minimal total losses around the lattice. The losses measurement is reproducible, faster and has lower noise compared to the lifetime measurement.

Once the environment is set up in Badger the user interface allows to simply determine subsets of knobs for the optimization and to chose whether to optimize lifetime, losses or both. The possibility to select knobs is very useful for

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO3A001

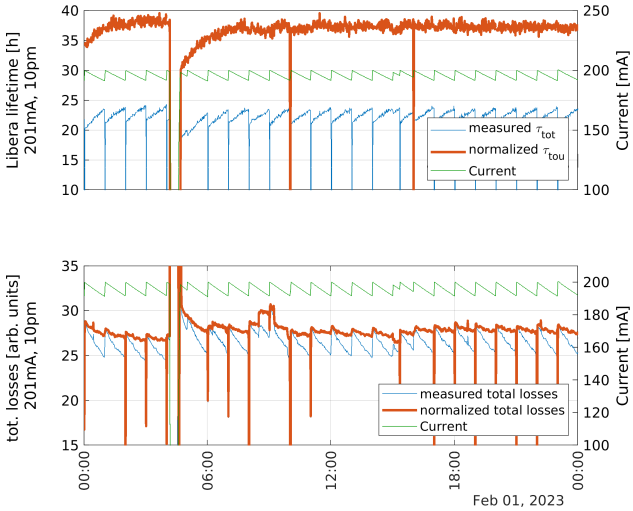

Figure 4: Measured current, total lifetime and total losses compared to normalized lifetime and total losses. One beam loss at sample ~250 evidences the Sokolov-Ternov polarization effect [24], not considered by the normalization.

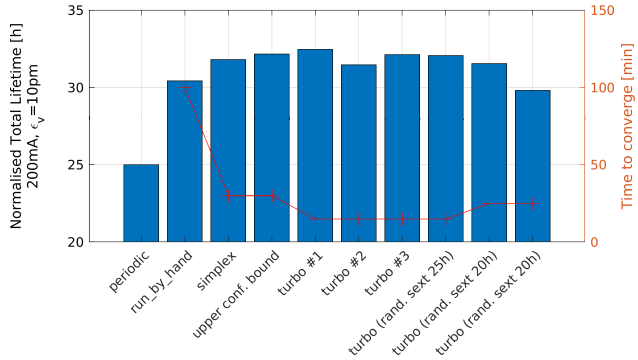

Figure 5: Total beam lifetime (Vacuum and Touschek lifetime contributions) and time to converge for different sextupoles and octupole settings. The label (periodic) corresponds to the absence of corrections on sextupole and octupoles. All other bars represent different optimizations performed. All values are normalized for total current and vertical emittance.

operational purposes, as it allows to reduce the number of variables without having to redefine a new environment.

Tests of the adaptation of Badger to the ESRF-SR were performed using the EBS simulator [21]. For this purpose two dummy devices were instantiated producing a false signal for losses and lifetime, proportional to the sextupoles' strengths. Starting from a random set of sextupoles, the optimizer was supposed to find the ground truth with all magnets set to zero. Only minimal debugging of interface and environments was necessary within the EBS-simulator to make Badger operational. No time was spent debugging during the few Machine Dedicated Time (MDT) slots allocated to the studies.

EBS Lifetime Optimization

Lifetime optimization is performed routinely for EBS, to recover the best possible lattice performances before User Service Mode (USM). The procedure for lifetime maximization is the following:

1. set all sextupole and octupole correctors to zero (periodic lattice)
2. kill any beam
3. cycle all magnets
4. refill uniform, 200 mA
5. open all insertion devices and collimators
6. set vertical scrapers to ±4 mm
7. correct tunes and start fast and slow orbit correction feedbacks
8. start 10 pm vertical emittance feedback
9. run sextupole and octupole knobs optimization
10. save setting, kill beam, cycle
11. refill uniform, 200 mA
12. measure Touschek and Vacuum Lifetime

At ESRF step 9 is usually done with the *run-by-hand* script [22]. This script scans the amplitude of each knob in a sequence, fits a parabola to the acquired data and keeps the amplitude producing the least total losses as measured by the 128 beam loss monitors around the SR [22]. With Badger/`Xopt` it was possible to easily replace this empiric sequential scan with more sophisticated algorithms.

The algorithms that were tested are:

- Nelder-Mead Simplex
- Upper Confidence Bound Bayesian Optimization
- Trust Region Bayesian Optimization (TuRBO)

Other algorithms such as RCDS [8] are available within Badger but there was no time to test them in control room.

The whole procedure described above is repeated identically for each algorithm. Figure 5 summarizes the results of the optimizations in terms of lifetime and time to converge. All optimizers tested improved the maximum lifetime, compared to the periodic settings and to the results of the standard optimization procedure. The empiric run-by-hand tuning takes about 100 min yielding about 30 h lifetime. The simplex and upper confidence bound algorithms take about 30 min before convergence and yield a lifetime of about 32 h. The TuRBO algorithm outperforms all other algorithms tested, by converging in only 15 min to total lifetime values above 32 h. Setting the sextupole and octupole strengths values found by the best run of optimizations in the storage ring led to the record Touschek lifetime values of $\tau_{tou} = 41 \pm 11$ h.

The total losses (and thus the lifetime) vary during the optimization process. Small amplitude of such variations towards high level of losses are desirable during the optimization process.

The least excursion of losses are observed for the TuRBO algorithm run within Badger.

The TuRBO algorithm was run three times in a row starting from the same initial sextupole and octupole setting. In all cases TuRBO converged to similar total beam lifetimes.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M03A001

To test the robustness of TuRBO random sextupoles were set in the lattice to disrupt the lifetime down to 20 h. Those settings give about 2-3 times larger sextupoles than the correction strengths necessary to optimize the lattice. Also in these conditions, TuRBO could recover lifetimes comparable to the best measured in operational conditions.

The TuRBO algorithm has been set in operation at ESRF thanks to Badger application. It allows to have faster optimizations and to reach larger final lifetimes with minimal sextupole strengths variations. It is fully transparent for users and can be thus used at any time during user service mode. This would allow to cope with possible impact of insertion devices and collimators settings on lifetime.

Set Up for PETRA III SR

As for the ESRF-EBS storage ring, a Badger *interface* is coded to act on the PETRA III TINE control system [25, 26] through its Python extension "PyTine". The commands used are `get_value`, `get_value_with_timestamp` and `set_value`. Two PETRA III environments were written specifically for the lifetime optimization experiment. The first one allows to act on the four skew quadrupoles QS1, QS2, QS3, QS4 which affect the machine global coupling (QS1 and QS4 for the difference resonance, QS2 and QS3 for the sum resonance). Also influencing the lifetime are the horizontal and vertical tunes. They are set by acting on the main quadrupoles in the FODO arc: QF and QD. For a more extended optimization the vertical dispersion in the damping wigglers is adjusted. This can be set through the second environment, where the vertical dispersion is corrected in the wiggler sections by varying the current in 8 skew quadrupoles located in the North and West ring sections. Similarly as in the EBS case, the environments also set the boundaries within the optimization algorithms can vary the magnet values, in order to avoid damages to the machine. The observable to be optimized also needs to be defined in the environment. For better accuracy in PETRA III the lifetime τ is computed by fitting the beam current $I(t)$ measured every second and averaged over 20 seconds with the following function:

$$I(t) = I(0) * e^{-\frac{t}{\tau}}$$

where $I(0)$ is the first measured value of the 20 seconds range. In order to take into account the natural lifetime reduction due to losses and reduce fluctuations the final optimized value normalized as described above in Eq. (1).

PETRA III Lifetime Optimization

The optimization performed on the PETRA III aimed at minimizing the linear coupling in the SR. To overcome limitations in the resolution of the beam size measurements, the beam lifetime is used as a proxy for vertical emittance and is thus minimized, as opposed to the EBS case. The Touschek lifetime is in fact proportional to the vertical beam size and thus to the vertical emittance. The current optimization routine in PETRA III evaluates the operational values of QS1-4 by moving the tunes to the $Q_x - Q_y$ difference resonance and trying to achieve the smallest tune difference as possible and is typically done a few times per year, mostly during startup after a shutdown. The set-points of the main quadrupoles QF and QD are manually tuned to achieve at the same time a good injection efficiency and a small horizontal emittance (near the design value of the natural emittance).The optimal currents for the skew quadrupoles in the North and West sections are computed from the inverse dispersion response matrix obtained by measurements. This process is usually performed every week. The experiment was performed in a dedicated machine development shift in March 2022 using a filling pattern of 10 bunches with a total current of 25 mA. Four different tests were performed acting on different sets of magnets. In the first case the lifetime was perturbed by setting the current in QS1, QS4 to 10 A and QS2, QS3 to 0 A. The currents in the same magnets were the optimization parameters. After observing that QS2 and QS3 had minimal impact on the lifetime a second test used QS1, QS4 together with the main quadrupoles QF and QD. For the third test the currents of the skew quadrupoles in the damping wigglers section was set to 0 to increase the lifetime and the optimization was performed initially using subsets of those quadrupoles (first the 4 in the West sector and then the 4 in the North sector). Lastly the optimization was performed tuning the 8 skew quadrupoles at the same time. In all of the aforementioned tests the optimization was performed using the BOtorch algorithm implemented in Badger and the procedure followed was:

1. The current in the magnets was modified till a consistent increase in the lifetime was observed.
2. The Badger optimization program was let to run till convergence.
3. The obtained lifetime was compared with the one of the magnets configuration at step 1 and with the operational values, in each case correcting the electrons orbit with the customary routine before acquiring the measurement.

The fourth and final test used the simplex algorithm to tune the eight skew quadrupoles starting the run from the operational values (instead of 0 as in the third test). No improvement was observed in the minimization, suggesting that the operational settings are optimal. Figure 6 shows the comparison between the initial, the operational and the optimized values for each of the tests described above. The Badger runs could generally find final settings corresponding to lifetime values comparable to the operational ones in all cases.

Injection Efficiency Optimization

Badger and `Xopt` are used also for the tuning of the transfer line magnetic elements between the booster (SY) and the EBS storage ring and to increase the dynamic aperture (DA) of the SR. Both tasks have as objective an increase of injection efficiency SY to SR (IE), bringing either improved

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M03A001

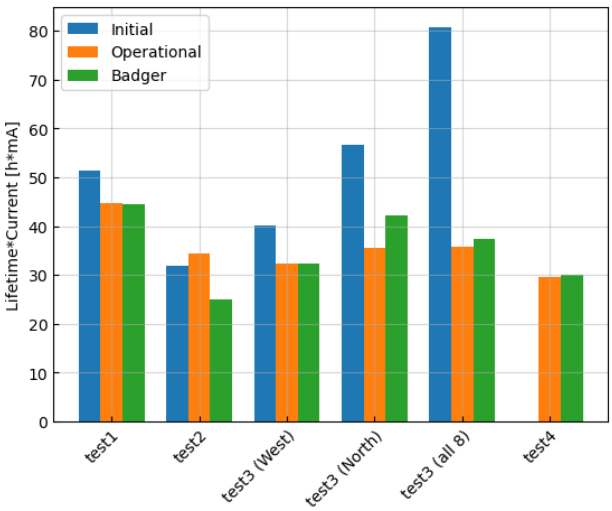

Figure 6: Comparison of the lifetime normalised with the current for the different tests performed in PETRA III. The blue bar represents the value obtained by de-tuning the relevant parameters from the operational value (in orange). The green bar is the value reached at the end of the Badger optimization. In the last test the optimization was performed from the operational values.

injected beam conditions or enlarged available space for injection. The IE is measured at EBS as the ratio between the sum signal on a BPM in the storage ring (calibrated to detect current variations) and the current at the extraction time in the booster. In order to measure such quantity the booster must be ramping and the injection in the SR triggered. A script is set up to trigger exactly ten shots of injection, only at the time of measurement. The average of these ten shots is the required measurement for each condition tested by the optimization process. This allows to: 1) minimize the time with injectors active for the sake of energy consumption and equipment ageing; and 2) reduce the total injected charge in the SR to avoid reaching the maximum stored beam current during an optimization (I_{max} = 200 mA). Figure 7 shows this process taking place.

For the optimization of the injected beam properties all the fourteen quadrupoles in the transfer line booster to storage ring (SY-SR) are varied within ±5 A of their set point. Four extraction and injection septa are also included as tuning parameters together with three vertical steerers, the extraction kicker amplitude, the RF phase between SY and SR and the time of extraction from the booster (extracted beam energy). For the optimization of the DA in the SR several sextupole and octupole knobs are selected as described above for the optimization of lifetime. In this case only the knobs with an effect on the DA (horizontal or vertical) and small impact on lifetime are kept. Seventeen such knobs are selected based on electron beam dynamics tracking simulations.

Figure 8 shows optimizations performed with the TuRBO optimizer within the Badger application.

The six optimizations visible in the figure correspond to:

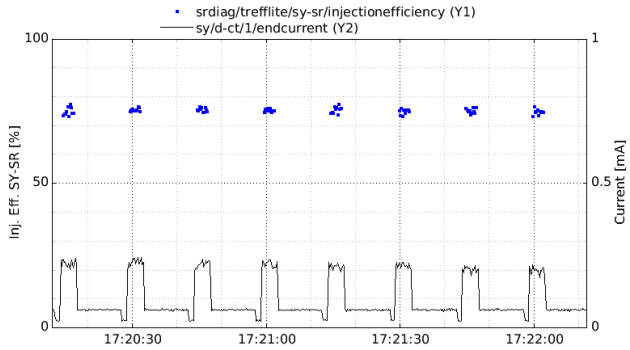

Figure 7: Injection efficiency SY-SR measured by ramping the booster and firing the electron beam gun only when needed. The black curve shows the current at the extraction time in the booster (endcurrent). The blue dots are measurements of IE.

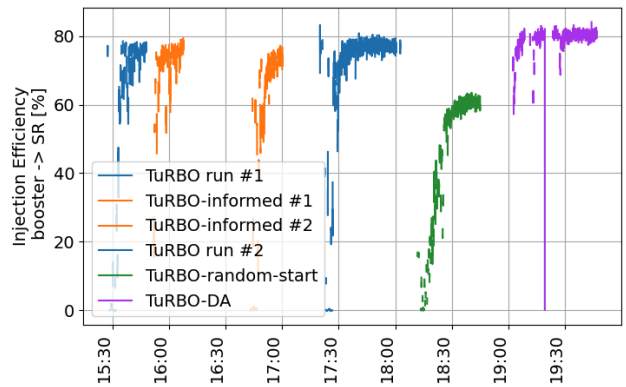

Figure 8: Injection efficiency and injected current during the optimizations performed with Badger and Xopt.

- optimization of the injected beam properties with TuRBO
- 2 runs of optimization of the injected beam properties with TuRBO starting from the previous run
- a second, longer optimization of the injected beam properties with TuRBO
- optimization of the injected beam properties with TuRBO starting from random quadrupole settings providing about 16 % injection efficiency as a starting point
- optimization of dynamic aperture with TuRBO

The optimizations of the injected beam properties finally resulted in ~1.5 % injection efficiency increase. The dynamic aperture optimizations contributed ~3.0 % to the total improvement reaching IE up to 82 %. The magnets set point variations observed are all reasonably small. Finally, Fig. 9 displays the lifetime and injection efficiency before and after the IE tuning.

The case of optimization started from random transfer line quadrupole settings stopped at about 60 %. In future

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-M03A001

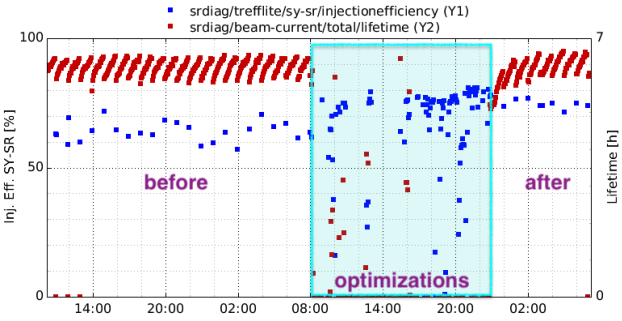

Figure 9: Injection efficiency and beam lifetime before and after the optimizations performed with Badger and Xopt.

optimization for this case other algorithms will be tests such as *upper confidence bound* or *simplex*.

No clear advantage is observed when running TuRBO starting with the data acquired from a precedent optimization run (#1 in Fig. 8). Further test will be run to confirm such observation.

Eventually, comparing to the previous user service mode settings (with closed insertion device gaps and safety collimators), an increase from 65 % to 75 % is observed and IE values appear to be more reproducible injection after injection with the new settings. The optimization of DA led to a small degradation of lifetime that was quickly recovered running Badger+Xopt(TuRBO) for lifetime optimization as described in the previous sections.

CONCLUSIONS

The Badger interface gives access to a variety of optimisers, including all the Xopt library algorithms. Badger provides a user friendly and versatile interface for SR optimizations. The result obtained during experimental tests on the EBS and PETRA III storage rings are in all cases at least comparable to those obtained by routine operation procedures and in most cases exceed the lifetime and injection efficiency values previously achieved with a very short optimization time.

Badger has been successfully used at EBS also for local optimization of sextupoles about a newly installed undulator and to optimize the lifetime after the installation of new magnets.

The Xopt TuRBO algorithm reached an optimal tuning configuration seven times faster than conventional scanning algorithms and more than fifty times faster than a human operator (assuming 2x8h shifts of manual tuning).

Badger has been set up at EBS with the Xopt TuRBO algorithm and is now available for operators to be used during user service mode for lifetime optimizations. In fact the optimization process is sufficiently fast and sensible (requires very small magnet setting variations), to be completely transparent for user operations (except the improved lifetime performances).

For PETRA III the use of Badger did not exceed the classic operation tuning and it is thus kept for machine studies only and as a valuable tool for the PETRA IV upgrade project [27].

Future use of Badger and Xopt will continue the studies started on injection efficiency optimization including additional parameters (such as the booster sextupoles at extraction) and testing different algorithms available in Xopt.

REFERENCES

[1] 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

[2] X. Huang, "Robust simplex algorithm for online optimization", *Phys. Rev. Accel. Beams*, vol. 21, no. 10, p. 104 601, 2018. doi:10.1103/PhysRevAccelBeams.21.104601

[3] B. Veglia *et al.*, "Extremum seeking for accelerator optimisation", English, in *Proc. IPAC'23*, Venice, Italy, 2023, pp. 372–375. doi:10.18429/JACoW-IPAC2023-MOPA137

[4] R. Roussel *et al.*, "Xopt: A simplified framework for optimization of accelerator problems using advanced algorithms", English, in *Proc. IPAC'23*, Venice, Italy, 2023, pp. 4847–4850. doi:10.18429/JACoW-IPAC2023-THPL164

[5] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace, 2009.

[6] R. Roussel *et al.*, "Turn-key constrained parameter space exploration for particle accelerators using Bayesian active learning", *Nat. Commun.*, vol. 12, no. 1, p. 5612, 2021. doi:10.1038/s41467-021-25757-3

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

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

[9] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design", *arXiv preprint*, 2009. doi:10.48550/arXiv.0912.3995

[10] D. Zhan and H. Xing, "Expected improvement for expensive optimization: A review", *J. Global Optim.*, pp. 1–38, 2020. doi:10.1007/s10898-020-00923-x

[11] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local Bayesian optimization", *Adv. Neural Inf Process. Syst.*, vol. 32, 2019. doi:10.48550/arXiv.1910.01739

[12] F. Irshad, S. Karsch, and A. Döpp, "Expected hypervolume improvement for simultaneous multi-objective and multi-fidelity optimization", *arXiv preprint*, 2021. doi:10.48550/arXiv.2112.13901

[13] 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

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO3A001

[14] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization", *Adv. Neural Inf Process. Syst.*, vol. 33, 2020. doi:10.48550/arXiv.2006.05078

[15] 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*, vol. 976, p. 164 273, 2020. doi:10.1016/j.nima.2020.164273

[16] Badger the optimizer, https://slac-ml.github.io/Badger/

[17] L. R. Dalesio, A. Kozubal, and M. Kraimer, "EPICS architecture", Los Alamos National Lab., NM (United States), Tech. Rep., 1991.

[18] G. Grygiel, O. Hensler, and K. Rehlich, "DOOCS: A distributed object oriented control system on PC's and workstations", in *PCaPAC conference*, 1996.

[19] P. Verdier, J. Pons, F. Poncet, and N. Leclercq, "Tango control system management tool", in *Proc. ICALEPCS*, 2011.

[20] Z. Zhang, *LCLS environment for Badger*, 2020. https://github.com/slaclab/Badger-Plugins/tree/master/environments/lcls

[21] S. M. Liuzzo *et al.*, "Update on the EBS Storage Ring Beam Dynamics Digital Twin", presented at ICALEPCS'23, Cape Town, South Africa, October 2023, paper THPDP010, this conference.

[22] N. Carmignani *et al.*, "Online Optimization of the ESRF-EBS Storage Ring Lifetime", in *Proc. IPAC'22*, Bangkok, Thailand, 2022, pp. 2552–2555. doi:10.18429/JACoW-IPAC2022-THPOPT001

[23] L. Torino and K. B. Scheidt, "New Beam Loss Detector System for EBS-ESRF", in *Proc. IBIC'18*, Shanghai, China, Sep. 2018, pp. 346–352. doi:10.18429/JACoW-IBIC2018-WEOB01

[24] A. Sokolov, I. Ternov, and C. Kilmister, *Radiation from Relativistic Electrons*. American Inst. of Physics, 1986. https://books.google.fr/books?id=cf8oAAAAYAAJ

[25] TINE website, http://tine.desy.de

[26] P. Bartkiewicz and P. Duval, "TINE as an accelerator control system at DESY", *Meas. Sci. Technol.*, vol. 18, no. 8, pp. 2379–2386, 2007. doi:10.1088/0957-0233/18/8/012

[27] I. Agapov *et al.*, "PETRA IV Storage Ring Design", in *Proc. IPAC'22*, Bangkok, Thailand, 2022, paper TUPOMS014, pp. 1431–1434. doi:10.18429/JACoW-IPAC2022-TUPOMS014