Original software publication

Geoff: The generic optimization framework & frontend for particle accelerator controls

P. Madysa [a] [ID],*, S. Appel [a] [ID], V. Kain [b] [ID], M. Schenk [b] [ID]

[a] *GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany*
[b] *CERN, European Organization for Nuclear Research, Geneva, Switzerland*

ARTICLE INFO

ABSTRACT

Geoff is a collection of Python packages that form a framework for automation of particle accelerator controls.

With particle accelerator laboratories around the world researching machine learning techniques to improve accelerator performance and uptime, a multitude of approaches and algorithms have emerged. The purpose of Geoff is to harmonize these approaches and to minimize friction when comparing or migrating between them. It provides standardized interfaces for optimization problems, utility functions to speed up development, and a reference GUI application that ties everything together.

Geoff is an open-source library developed at CERN and maintained and updated in collaboration between CERN and GSI as part of the EURO-LABS project. This paper gives an overview over Geoff's design, features, and current usage.

Code metadata

Current code version	0.17.11
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00339
Permanent link to Reproducible Capsule	none
Legal Code License	GPL-3.0-or-later OR EUPL-1.2+
Code versioning system used	Git
Software code languages, tools, and services used	Gitlab, Python, PyTorch, Ruff
Compilation requirements, operating environments & dependencies	Python 3.9+; NumPy, Gymnasium; the GUI application depends on CERN-internal components
If available Link to developer documentation/manual	https://cernml-coi.docs.cern.ch/
Support email for questions	geoff-community@cern.ch

1. Motivation and significance

The field of *accelerator controls* is concerned with the development of hardware and software for the operation of a particle accelerator or a complex of accelerators. It presents a number of challenges:

1. The hardware is operated *around the clock* and subjected to considerable levels of radioactivity.
2. *Strong uptime requirements* mean that the software must be efficient, reliable and deterministic even in case of failures.
3. Manpower, funds and *upgrade efforts* are limited, even though accelerators operate for years and decades.

Faced with this, software engineers responsible for building and maintaining controls systems used to eschew the latest technologies or dynamic programming languages, and favored more traditional and battle-tested technologies instead. In practice, this means that middleware and control-room applications used the Java language and ecosystem. This decision has traded performance for reliability [1].

At the same time, the field of machine learning (ML) can no longer be ignored to satisfy the performance and uptime requirements with increasing accelerator complexity [2]. Early attempts to use ML for accelerator controls [3,4] were successful in individual test cases, but did not gain traction due to the limited ML ecosystem in Java. Most

* Corresponding author.
E-mail addresses: p.madysa@gsi.de (P. Madysa), s.appel@gsi.de (S. Appel), verena.kain@cern.ch (V. Kain), michael.schenk@cern.ch (M. Schenk).

contemporary ML research is based on the Python language and frameworks like TensorFlow and PyTorch.

While individual accelerator scientists have used these technologies on a case-by-case basis [5], this has led to a lot of *duplicated work, lack of compatibility* and *disregard for usability*. This hampers adoption and makes comparison of the different approaches much more difficult [6].

Geoff [7] is a Python framework developed at CERN and open-sourced as part of the EURO-LABS [8] project. It provides:

- a *standardized API* for numerical optimization and reinforcement learning (RL), making it easier to share and compare algorithms;

- a *library of utility functions* to avoid duplicate work;
- and a *GUI application* to easily |deploy solutions in the accelerator control room.

Geoff focuses on the execution of optimization algorithms and trained RL agents. Tasks associated with the process of RL such as model storage and benchmarking are beyond its scope.

Because Geoff is written in Python [9], the full ecosystem of the latest ML is available. In this way, it bridges the gaps between ML experts, control-room operators, and accelerator physicists.

In a typical case, Geoff is deployed at a research institute by a group of software and IT infrastructure experts, e.g. the institute's controls department. Operators—engineers managing the day-to-day operations of the accelerator from a control room—use the Geoff application to solve certain well-defined *optimization problems*. These problems are Python plugins written by domain experts with knowledge of specific accelerator subsystems—often physicists or engineers.

Geoff's open nature enables these groups to communicate and empowers operators to improve existing and create new plugins. At the same time, its emphasis on versioning ensures that any erroneous updates can be easily rolled back.

Geoff is preceded by a number of similar frameworks [10–12]. It improves upon them with its radical approach to extensibility, making many problems solvable that would otherwise be impossible to express.

It competes [13] with Xopt [14] and Badger [15]. Both have distinct advantages, but lack support for RL or versioning and dependency management.

Geoff's API builds on that of the Gymnasium [16] library. By default, its application is bundled with state-of-the-art algorithms, such as provided by Stable-Baselines3 [17], SciPy [18] and Py-BOBYQA [19, 20]. It uses NumPy [21] for data transfer and PyQtGraph [22] for visualization. Plugins can visualize their own data via Matplotlib [23].

Geoff is hosted at CERN and its DevOps automation is facilitated by CERN's Acc-Py project [24].

2. Software description

Geoff consists of a number of Python packages, some more central to its concepts than others. The most important ones are as follows:

cernml-coi defines and standardizes the **C**ommon **O**ptimization **I**nterfaces interfaces for optimization problems. The interfaces are divided by capabilities so that an optimization problem can support numerical optimization, RL, or both. Interfaces for RL are provided by the Gymnasium package, which this package builds upon.

Various mix-in interfaces can express additional capabilities, e.g. configurability via the GUI application. The package also provides routines that verify whether an implementation satisfies the invariants of these interfaces.

cernml-coi-optimizers standardizes the interface specifically for numerical optimization algorithms. It also provides adapters to this interface for many popular libraries such as SciPy. The adapters are written as optional dependencies, so that they need not be installed if not required.
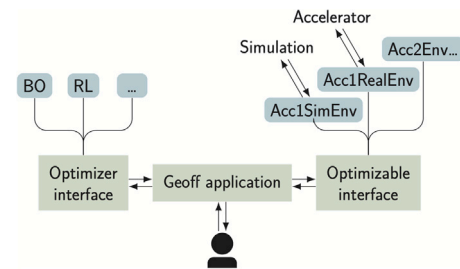

Fig. 1. Design architecture of *Geoff*.

cernml-coi-utils provides a number of utility functions and classes for optimization problems. These allow Geoff plugins to remain short and focused on the task at hand. Examples include a queue for simplified communications with an accelerator device and a method decorator to manage the state of a custom Matplotlib visualization.

geoff-app is a reference implementation of a GUI application based on PyQt [25] that loads a set of plugins and communicates with them via the above interfaces. Research institutes outside of CERN are encouraged to fork this project to take local concerns into account and provide accelerator-specific information. See Section 3 for an example of how this is realized at CERN.

All of these packages follow Semantic Versioning [26] independently. This allows them to evolve at different speeds and keeps version numbers meaningful; e.g. a backwards-incompatible change to the utilities library (incrementing its MAJOR version number) has less impact than a similar change to the interfaces.

2.1. Software architecture

The general architecture of the framework is shown in Fig. 1. The user controls the system via a *host application*. This may be a GUI application like **geoff-app**, or a console script, or even a server endpoint.

The application loads available algorithms and optimization problems, either by direct import or via Python's *entry points* API [27]. In addition to the *generic* algorithms, *custom* algorithms specific to a selected optimization problem may be loaded via *custom-optimizer providers* (for numerical optimization) and *custom-policy providers* (for RL).

These providers are classes or functions that are each tied to one specific optimization problem and may define additional, custom-tailored algorithms. This is useful in the case of Bayesian optimization (BO), where complicated kernel functions must often be defined in a way that makes sense only for a specific problem. The host application can load these providers in two ways: either via an entry point, or via the optimization problem class if it implements the provider interfaces itself. This makes it easy to publish additional algorithms without coordinating with whoever maintains the host application.

Once the user chooses an algorithm to use and a problem to solve, both are loaded dynamically via Python's import system, and the application communicates with them through the appropriate interfaces. In particular, the application drives the interaction loop between optimization problem and algorithm, transmits data between them, and visualizes intermediate results to the user.

The optimization problem class, in turn, communicates with the controls system of the particle accelerator; i.e. it drives changed settings to the devices near the accelerator, and receives monitoring data from other devices. Authentication of the user and authorization of the settings change are generally handled by the controls system itself [28]. Geoff's architecture also allows the class to do anything *besides* settings

changes, including starting subprocesses and communicating |with a simulation of the accelerator.

The interfaces defined by **cernml-coi-optimizers** supports *closed-loop* implementations of optimization algorithms. Such implementations typically only provide a `solve` function that, when called, executes the *entire* algorithm and does not yield the control flow until the algorithm is completed.

In contrast, other frameworks like Xopt [14] require an *open-loop* implementation. Such implementations provide a state machine with methods sometimes called `ask` and `tell`. When `ask` is called, it returns a candidate point x to evaluate; `tell` is called with the value $f(x)$ of the objective function at that point. Both methods should be called in alternation until the algorithm terminates.

The difference between open and closed loops is significant because most implementations of numerical optimization algorithms have a closed loop. While it is trivial to translate an open loop to a closed loop, the reverse is usually *impossible* without significantly rewriting the algorithm. Admitting the more general case of closed-loop optimizers allows Geoff to support a wider range of algorithms than comparable frameworks. BOBYQA [19,20] is one example of an algorithm for which no open-loop implementation has been published yet.

2.2. Software functionalities

Geoff reduces the complexity of making m optimization problems compatible with n optimization algorithms from $\mathcal{O}(m \cdot n)$ to $\mathcal{O}(m + n)$. The advantage is two-fold: For one, a newly developed algorithm can be used immediately on any existing problem, speeding up its adoption. For another, a given problem can be solved with a wide range of different algorithms, depending on situational context and evolving expertise of operators [29].

At all points, |plugin developers have full freedom to use either the high-level abstractions of the Geoff utilities library, or to directly use the features of the underlying controls system. This allows plugins to solve not only simple toy problems, but also more complex ones, where e.g. an accelerator device is known to behave in an unusual fashion but it is not feasible to fix the issue at the source [29].

Because plugins are *independent packages* with their own dependency declarations, they can scale from minimal proof-of-concept implementations to complex state machines that call out to subprocesses or request data from the accelerator's monitoring devices. Because plugins have their own versioning scheme, faulty upgrades are trivial to roll back without excessive downtime in the accelerator.

The dynamic nature of the plugin architecture also allows plugin developers to test their code using a deployed version of the host application, and include it in a future one. The modular architecture of Geoff also means that plugin developers do not have to use the deployed application at all, and instead e.g. drive their plugin via a console script. This can reduce turnaround time in the early development cycle.

2.3. Sample code snippets analysis

This section shows an example implementation of a Geoff plugin, adopted from the developer documentation. It adjusts a set of *corrector magnets* in a section of the AWAKE accelerator at CERN [30,31]. An electron beam traveling through the beam line is measured via a set of *beam position monitors*. The objective of the optimization problem is to adjust the correctors such that the beam travels as closely to the central axis of the monitors as possible.

The first lines import a number of packages (Gymnasium [16], NumPy [21], Matplotlib [23]) and the *Common Optimization Interfaces* of Geoff. PyJapc is a CERN-internal package to communicate with the controls system.

```
1  import gymnasium as gym
2  import numpy as np
3  from cernml import coi
4  from gymnasium.spaces import Box
5  from matplotlib import pyplot as plt
6  from pyjapc import PyJapc
```

Geoff represents optimization problems as Python classes. They inherit from one or more of the standard interfaces, depending on whether they support single-objective numerical optimization (represented by the abstract base class `SingleOptimizable`), RL (represented by `Env`), or both.

```
9  class AwakeSteering(coi.SingleOptimizable, gym.Env):
```

All classes contain a class-scope dictionary called `metadata`. It contains information that a host application can use to infer how to interact with the plugin. Some dictionary keys are standardized by Gymnasium and some by Geoff. Geoff's developer documentation describes all standard keys. Laboratories are also encouraged to standardize their own, appropriately prefixed keys. All keys are considered optional and have reasonable default values.

```
10     metadata = {
11         "render_modes": ["human"],
12         "cern.machine": coi.Machine.AWAKE,
13         "cern.japc": True,
14     }
```

Here, the list behind `render_modes` specifies the ways in which the plugin state can be visualized. Gymnasium defines the mode "`human`" for interactive visualization, typically initiated without a host application. We implement this via Matplotlib, but other libraries can be used. The other keys are laboratory-specific: "`cern.machine`" marks the relevant accelerator in the CERN complex. CERN's host application filters the optimization problems presented to the user based on this value. If "`cern.japc`" is set, CERN's host application handles user authentication for the plugin and passes the configured PyJapc object to its `__init__` method.

```
16     CORRECTORS = ["logical.RCIBH.430029/K", ...]
17     MONITORS = ["TT43.BPM.430028/Acquisition", ...]
```

These lines are not part of the Geoff interface. They list the *device names* that will be passed to PyJapc to communicate with the correct devices on the accelerator. They have been abbreviated here for brevity.

The following lines define the mathematical *domain* of the algorithm: this is `optimization_space` for numerical optimization and `action_space` for RL. RL algorithms additionally require the domain of possible observations for normalization purposes (`observation_space`).

```
19     action_space = Box(-0.1, 0.1, shape=[len(CORRECTORS)])
20     observation_space = Box(-1.0, 1.0, [2 * len(MONITORS)])
21     optimization_space = Box(-1.0, 1.0, [len(CORRECTORS)])
```

Domains must be subclasses of `gymnasium.Space`. We use `Box` here, which represents a multidimensional rectangular bounding box. While `Box` is the most common type of space, discrete and composite spaces exist as well.

Next, an initializer stores the chosen render mode and uses private helper methods to fetch the state of the machine from the controls system:

```
23     def __init__(self, japc=None, render_mode=None):
24         self.render_mode = render_mode
25         self.japc = japc or PyJapc()
26         self.latest_readings = self._recv_bpm_readings()
27         self.correctors = self._recv_corrector_values()
```

These helper methods are defined directly below. They are the only part of the class that directly interacts with the controls system:

```
29        def _recv_bpm_readings(self):
30            props = [self.japc.getParam(m) for m in self.MONITORS]
31            return np.array(
32                [[p["horPos"], p["verPos"]] for p in props]
33            ).flatten()
34
35        def _recv_corrector_values(self):
36            return np.array([
37                self.japc.getParam(c) for c in self.CORRECTORS
38            ])
39
40        def _send_corrector_values(self, values):
41            self.correctors = values.copy()
42            for addr, value in zip(self.CORRECTORS, values):
43                self.japc.setParam(addr, value)
```

Next comes the implementation of the interface methods for RL and numerical optimization. Both interfaces require two methods. First, a method that initializes either an RL episode or an optimization run (`reset` and `get_initial_params` resp.):

```
45        def reset(self, *, seed=None, options=None):
46            super().reset(seed=seed, options=options)
47            correctors = self.optimization_space.sample()
48            self._send_corrector_values(correctors)
49            info = {}
50            if self.render_mode == "human": self.render()
51            return self.correctors.copy(), info
52
53        def get_initial_params(self, *, seed=None, options=None):
54            params, _info = self.reset(seed=seed, options=options)
55            return params
```

Secondly, a method that performs a step of the algorithm and returns new data (`compute_single_objective` for optimization, `step` for RL):

```
57        def compute_single_objective(self, params):
58            self._send_corrector_values(params)
59            self.latest_readings = self._recv_bpm_readings()
60            if self.render_mode == "human": self.render()
61            return np.sqrt(np.mean(self.latest_readings**2))
62
63        def step(self, action):
64            correctors = self.correctors + action
65            reward = -self.compute_single_objective(correctors)
66            obs = self.latest_readings.copy()
67            terminated = reward > -0.1
68            truncated = False
69            info = {}
70            return obs, reward, terminated, truncated, info
```

All methods *also* perform a rendering step *if* the render mode is "`human`". This is required by Gymnasium's definition of the render mode.

Both interfaces define default attributes and methods that can be overridden for customization. For example, `param_names` allows `SingleOptimizable` to attach names to the individual axes of its `optimization_space`:

```
71        param_names = CORRECTORS
```

Another example is the `render` method, which implements visualization:

```
73        def render(self):
74            if self.render_mode != "human":
75                return super().render()
76            figure = plt.figure("AwakeElectronBeamSteering")
77            figure.clear()
78            plt.plot(self.latest_readings, "|-")
79            plt.xlabel("BPM")
80            plt.ylabel("Beam position (mm)")
81            figure.show(warn=False)
82            figure.canvas.draw_idle()
83            return None
```

Finally, the plugin registers itself with Geoff:

```
86    coi.register("AwakeSteering-v1", entry_point=AwakeSteering)
```

This ensures that the plugin will be found when a host application aggregates all available plugins. While registration occurs here at runtime, Geoff also supports entry points for registration at installation.

3. Illustrative examples

Fig. 2 shows a screenshot of the Geoff GUI application that is deployed in the CERN control room. It shows the application state after a successful run of the BOBYQA algorithm.

The optimization problem here was to maximize the intensity of the particle beam injected into the Proton Synchrotron (PS), a particle accelerator at CERN. The main difficulty lies in the communication with devices in different *timing domains* (i.e. with unaligned data publishing rates). Data from these devices must potentially be polled multiple times per iteration of the optimization algorithm and carefully synchronized. This would be impossible to implement in a framework where data acquisition is the framework's responsibility rather than the plugin's.

The central area of the window (e) is taken up by four sub-windows that show live updates during optimization:

- The top windows show the status of the most recent step; they are provided by the plugin itself.
- The bottom windows show the progress of optimization; they are always provided by the host application.

The optimization control area (d) allows the user to choose between numerical optimization, training an RL agent and executing an RL agent that has already been trained. The user can also start, cancel and revert these procedures here. This area is also where the user selects both the problem to solve and the algorithm to use.

The application window shown here also contains CERN-specific GUI elements such as:

- the current schedule of acceleration cycles (a);
- a log-in button that may be required when modifying mission-critical settings (b);
- a selector for the acceleration cycle whose settings are modified (c);
- an expandable status line with a log of important events (f).

These elements are maintained by the CERN-internal Acc-Py project [24]. Other research institutes are expected to develop GUI elements that are relevant to their respective operations.

While Geoff supports a GUI host application, it does not require one. As an example, when porting efforts at GSI began, no GUI application without CERN dependencies existed. Nonetheless, the framework could be used on a command-line terminal. Fig. 3 shows a screenshot of such an optimization run at GSI. Logging messages were being printed in the terminal, while the optimization progress was visualized via Matplotlib and X11 window forwarding.

4. Impact

Geoff has enabled studies where a comparison of numerical optimization and RL is of interest [32] and where the complex optimization procedures excluded less flexible frameworks [33].

It has improved the pursuit of other research questions by making it easier to share optimization algorithms [34]; facilitating the migration to better algorithms [29]; and reducing the time from proof of concept to operational deployment [35].

It is used extensively in the daily operations of the LINAC4 linear accelerator and the PS Booster [36], the PS [37,38], and the Low-Energy Ion Ring (LEIR) [39–41]. LEIR particularly benefits from Geoff because it has no dedicated operators and most tasks used to be done manually.

At the Super Proton Synchrotron (SPS), Geoff reduced the positioning time of the slow-extraction septa from eight hours to ten minutes [43, p. 9]. It was also used in the compensation of main-power converter noise from the electrical grid and allowed the continuous
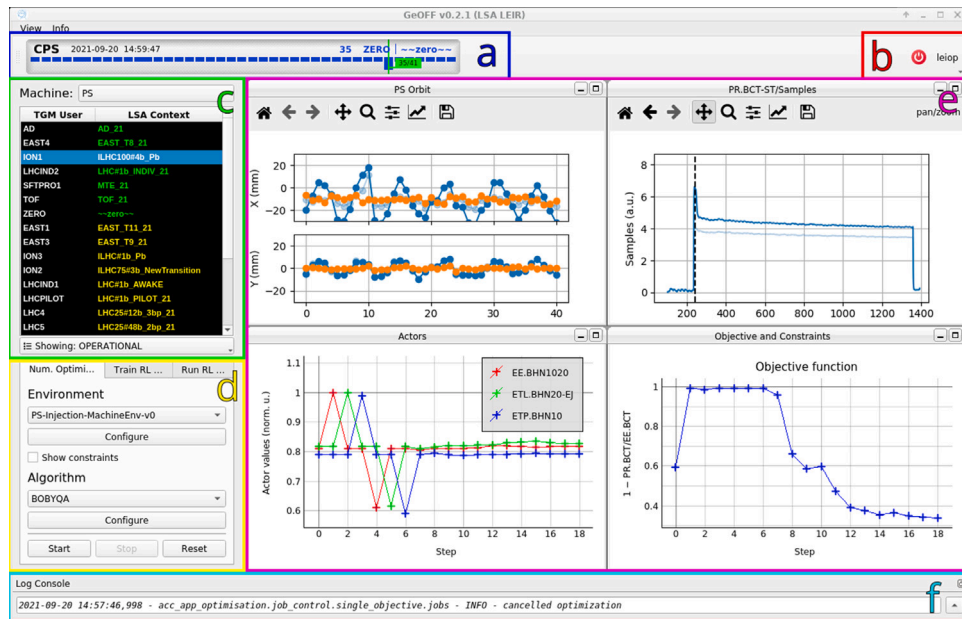
Fig. 2. Example of the Geoff GUI application deployed in the CERN control room. The various GUI components are framed in several colors and labeled a–f. See the text for their description.
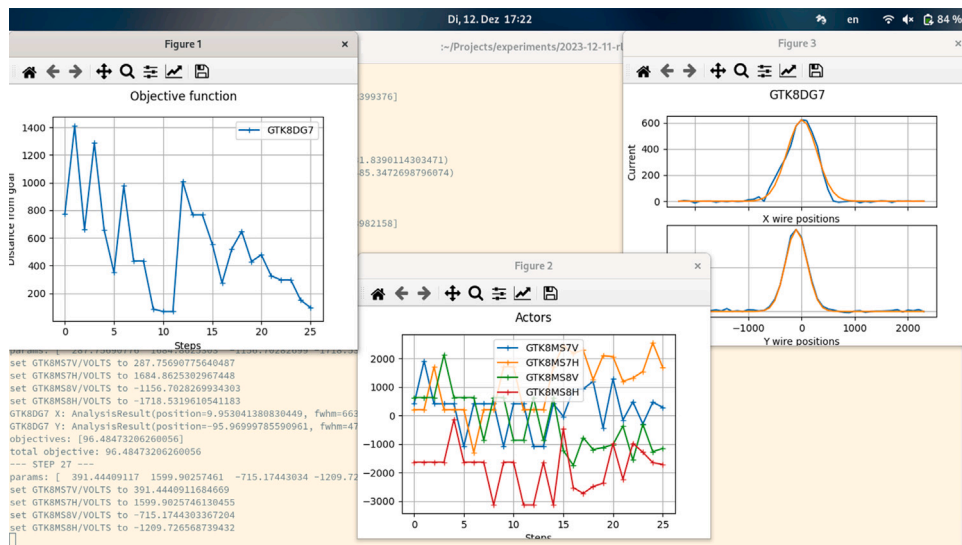

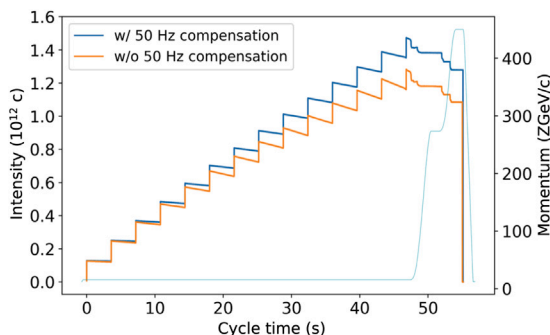**Fig. 3.** Example of Geoff being run in a terminal in the GSI control room.

Fig. 4. Intensity of the beam injected from the SPS into the LHC with and without the Bayesian optimization of main-power converter noise [42].

upgrade of algorithms over multiple years; the latest one (based on BO), improves ion beam transmission to the Large Hadron Collider (LHC) by 15–20 % [42], as Fig. 4 shows. Finally, Geoff facilitated the operational deployment of simulation-trained RL agents for beam trajectory correction in the transfer lines from SPS to the North Experimental Area [44].

At the source of ion beams for the LINAC3 linear accelerator, Geoff is used to optimize and continuously tune various parameters to provide a high and stable output current for ion operations [45]. The problem is a mix of slow and fast dynamics and the nested-loop structure used to solve it would be difficult to implement without Geoff's flexibility.

Geoff is used at every accelerator of the CERN complex except the LHC [43,46,47], which was designed from the start in a way that avoids black-box optimization and instead prefers non-iterative, model-based algorithms.
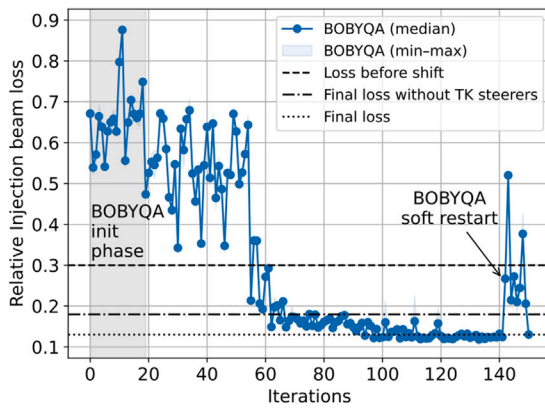
Fig. 5. Automatic online steering of the SIS18 multi-turn injection using the BOBYQA optimization algorithm [48]. The gray shaded area marks the initialization phase of the algorithm. The final point presents an additional evaluation at the found optimum; it is not strictly part of the algorithm.

Geoff is also being deployed at GSI [13]. Successful tests have been performed in simulation [49] and on real accelerators. The beam loss at injection into the SIS18 synchrotron could be reduced from 40 % to 15 %, as Fig. 5 shows [48]. At GSI's Fragment Separator (FRS), Geoff has been used to adjust the ion beam trajectory. Its flexibility was crucial due to the large number of interacting components [50].

Finally, as part of EURO-LABS [8], efforts are ongoing to use Geoff at the laser facility at CEA Paris-Saclay. Preliminary results are expected at the end of 2025.

Commercial use of Geoff is currently not foreseen, given its large focus on an academic context. However, such use is possible and permitted by its open-source license.

5. Conclusions

Geoff successfully unified previously uncoordinated efforts to use ML for accelerator control automation at CERN. It is used throughout the CERN complex; this ensures maintenance and further development for years to come. At the same time, its support by EURO-LABS made it possible to make it open-source and available to other research institutes.

Future plans for the software are outlined in a roadmap in the developer documentation (see). One such plan is the inclusion of support for multi-objective optimization[51] via a framework like PyMoo [52]. Support for this branch of optimization would greatly expand the scope in which Geoff can be used. Another plan is to provide tools to allow bootstrapping a Bayesian optimizer with previously recorded data to greatly speed up experimentation with such algorithms. Finally, we intend to provide a redesigned reference GUI application with support for data collection and export to encourage use by non-experts in a wider range of accelerator laboratories.

CRediT authorship contribution statement

P. Madysa: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Investigation. **S. Appel:** Writing – review & editing, Visualization, Validation, Resources, Project administration, Funding acquisition. **V. Kain:** Writing – review & editing, Visualization, Validation, Software, Investigation, Conceptualization. **M. Schenk:** Writing – review & editing, Visualization, Validation, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Govindan V, Jayaprakash J, Park C, Lee JR, Cangul IN. Optimization-based design and control of dynamic systems. Babylon J Math 2023;2023:30–5. http://dx.doi.org/10.58496/BJM/2023/006.

[2] Edelen A, et al. Opportunities in machine learning for particle accelerators. 2018, arXiv:1811.03172.

[3] Huang X, Safranek J. Online optimization of storage ring nonlinear beam dynamics. Phys Rev ST Accel Beams 2015;18(8):084001. http://dx.doi.org/10.1103/PhysRevSTAB.18.084001, arXiv:1502.07799.

[4] Appel S, Geithner W, Reimann S, Sapinski M, Singh R, Vilsmeier D. Application of nature-inspired optimization algorithms and machine learning for heavy-ion synchrotrons. Internat J Modern Phys A 2019;34(36):1942019. http://dx.doi.org/10.1142/S0217751X19420193.

[5] Kain V, Hirlander S, Goddard B, Velotti FM, Zevi Della Porta G, Bruchon N, et al. Sample-efficient reinforcement learning for CERN accelerator control. Phys Rev Accel Beams 2020;23(12):124801. http://dx.doi.org/10.1103/PhysRevAccelBeams.23.124801.

[6] Kim HS, Park C, Govindan V, Vasudevan S, Kumar GJ. Systematic reliability optimization (ASRO). Babylon J Math 2023;2023:50–3. http://dx.doi.org/10.58496/BJM/2023/010.

[7] Madysa P, Kain V. Geoff—the generic optimization framework and frontend. Software 2025. http://dx.doi.org/10.5281/zenodo.8434512.

[8] Colonna M, Navin A. EURO-LABS: Europe's super community of sub-atomic researchers. Nucl Phys News 2023;33(2):3–4. http://dx.doi.org/10.1080/10619127.2023.2198906.

[9] Python Software Foundation. Python language reference. Software 1990–2023. [online] https://www.python.org/.

[10] Agapov I, Geloni G, Tomin S, Zagorodnov I. OCELOT: A software framework for synchrotron light source and FEL studies. NIM A 2014;768:151–6. http://dx.doi.org/10.1016/j.nima.2014.09.057.

[11] Piselli E, Akroh A. New CERN Proton Synchrotron beam optimization tool. In: Proc. ICALEPCS'17. Geneva, Switzerland: JACoW; 2018, p. 692–6. http://dx.doi.org/10.18429/JACoW-ICALEPCS2017-TUPHA120.

[12] Geithner W, et al. Genetic algorithms for machine optimization in the FAIR control system environment. In: Proc. IPAC'18, no. 9 in international particle accelerator conference. Geneva, Switzerland: JACoW Publishing; 2018, p. 4712–5. http://dx.doi.org/10.18429/JACoW-IPAC2018-THPML028.

[13] Roussel R, et al. Bayesian optimization algorithms for accelerator physics. Phys Rev Accel Beams 2024;27(8):084801. http://dx.doi.org/10.1103/PhysRevAccelBeams.27.084801, arXiv:2312.05667.

[14] Roussel R, Edelen A, Bartnik A, Mayes C. Xopt: A simplified framework for optimization of accelerator problems using advanced algorithms. In: Proc. IPAC'23. Geneva, Switzerland: JACoW Publishing; 2023, p. 4796–9. http://dx.doi.org/10.18429/jacow-ipac2023-thpl164.

[15] Zhang Z, Böse M, Edelen A, Garrahan J, Hidaka Y, Mayes C, et al. Badger: The missing optimizer in ACR. In: Proc. IPAC'22. Geneva, Switzerland: JACoW Publishing; 2022, p. 999–1002. http://dx.doi.org/10.18429/JACoW-IPAC2022-TUPOST058.

[16] Towers M, Kwiatkowski A, Terry J, Balis JU, Cola GD, Deleu T, et al. Gymnasium: A standard interface for reinforcement learning environments. 2024, arXiv:2407.17032.

[17] Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N. Stable-Baselines3: Reliable reinforcement learning implementations. J Mach Learn Res 2021;22:1–8. URL http://jmlr.org/papers/v22/20-1364.html.

[18] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental algorithms for scientific computing in python. Nature Methods 2020;17:261–72. http://dx.doi.org/10.1038/s41592-019-0686-2.

[19] Cartis C, Fiala J, Marteau B, Roberts L. Improving the flexibility and robustness of model-based derivative-free optimization solvers. ACM Trans Math Software 2019;45. http://dx.doi.org/10.1145/3338517.

[20] Coralia Cartis LR, Sheridan-Methven O. Escaping local minima with local derivative-free methods: a numerical investigation. Optimization 2022;71:2343–73. http://dx.doi.org/10.1080/02331934.2021.1883015.

[21] Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature 2020;585:357–62. http://dx.doi.org/10.1038/s41586-020-2649-2.

[22] Moore O, Jessurun N, Nemitz N, Chase M, Campagnola L. PyQtGraph - high performance visualization for all platforms. 2023, http://dx.doi.org/10.25080/gerudo-f2bc6f59-01d.

[23] Hunter JD. Matplotlib: A 2D graphics environment. Comput Sci Eng 2007;9:90–5. http://dx.doi.org/10.1109/MCSE.2007.55.

[24] Elson P, Baldi C, Sinkarenko I. Introducing Python as a supported language for accelerator controls at CERN. In: Proc. ICALEPCS'21. Geneva, Switzerland: JACoW Publishing; 2022, p. 236–41. http://dx.doi.org/10.18429/JACoW-ICALEPCS2021-MOPV040.

[25] The Qt Company Ltd. Qt for Python. Software 2022. [online]. https://doc.qt.io/qtforpython-5/.

[26] T. Preston-Werner. Semantic versioning, [online].https://semver.org/.

[27] Python Packaging Authority. Entry points specification, [online]. https://packaging.python.org/en/latest/specifications/entry-points/.

[28] Yastrebov I. Role-Based Access Control for the Large Hadron Collider at CERN. Int J Comput Commun Control 2010;5(3):398–409.

[29] Kain V, Effinger E, Follin F, Velotti F, Fraser M, Schenk M, et al. Slow extracted spill ripple control in the CERN SPS using adaptive Bayesian optimisation. In: Proc. IPAC'24. Geneva, Switzerland: JACoW Publishing; 2024, p. 1790–3. http://dx.doi.org/10.18429/JACoW-IPAC2024-TUPS55.

[30] Adli E, et al. Acceleration of electrons in the plasma wakefield of a proton bunch. Nature 2018;561(7723):363–7. http://dx.doi.org/10.1038/s41586-018-0485-4.

[31] Kain V, Hirlander S, Goddard B, Velotti FM, Della Porta GZ, Bruchon N, et al. Sample-efficient reinforcement learning for CERN accelerator control. Phys Rev Accel Beams 2020;23:124801. http://dx.doi.org/10.1103/PhysRevAccelBeams.23.124801.

[32] Madysa N, Alemany-Fernández R, Biancacci N, Goddard B, Kain V, Velotti F. Automated intensity optimisation using reinforcement learning at LEIR. In: Proc. IPAC'22. Geneva, Switzerland: JACoW Publishing; 2022, p. 941–4. http://dx.doi.org/10.18429/JACoW-IPAC2022-TUPOST040.

[33] Bruchon N, Karpov I, Madysa N, Papotti G, Quartullo D. Operational tool for automatic setup of controlled longitudinal emittance blow-up in the CERN SPS. In: Proc. 19th int. conf. accel. large exp. phys. control syst.. Geneva, Switzerland: JACoW Publishing; 2024, p. 723–8. http://dx.doi.org/10.18429/JACoW-ICALEPCS2023-TUPDP086.

[34] Uden C, Huschauer A, Pahl H, Schenk M, Madysa N, Kain V. Multi-objective extremum seeking to control drifts in the transverse beam splitting efficiency of the multi-turn extraction at the CERN proton synchrotron. In: Proc. IPAC'23. Geneva, Switzerland: JACoW Publishing; 2023, p. 1675–8. http://dx.doi.org/10.18429/JACoW-IPAC2023-TUPA159.

[35] Fraser M, Coly M, Guerrero A, Huschauer A, Jensen S, Levasseur S, et al. Matching studies between the CERN PSB and PS using turn-by-turn beam profile acquisitions with a residual beam gas ionisation monitor. In: Proc. IPAC'22. Geneva, Switzerland: JACoW Publishing; 2022, p. 2161–4. http://dx.doi.org/10.18429/JACoW-IPAC2022-WEPOTK043.

[36] Skowroński P, Fraser M, Vojskovic I. Experience with computer-aided optimizations in LINAC4 and PSB at CERN. In: Proc. IPAC'22. Geneva, Switzerland: JACoW Publishing; 2022, p. 945–8. http://dx.doi.org/10.18429/JACoW-IPAC2022-TUPOST041.

[37] Huschauer A, Coly M, Cotte D, Damerau H, Delrieux M, Dumont J-C, et al. Beam commissioning and optimisation in the CERN Proton Synchrotron after the upgrade of the LHC injectors. In: Proc. IPAC'22. Geneva, Switzerland: JACoW Publishing; 2022, p. 54–7. http://dx.doi.org/10.18429/JACoW-IPAC2022-MOPOST006.

[38] Huschauer A, Coly M, Cotte D, Damerau H, Delrieux M, Dumont J, et al. Beam performance and operational efficiency at the CERN proton synchrotron. In: Proc. IPAC'23. Geneva, Switzerland: JACoW Publishing; 2023, p. 1671–4. http://dx.doi.org/10.18429/JACoW-IPAC2023-TUPA158.

[39] Biancacci N, Albright S, Alemany-Fernández R, Alves D, Angoletta M, Barrientos D, et al. LINAC3, LEIR and PS performance with ions in 2021 and prospects for 2022. In: Proc. IPAC'22. Geneva, Switzerland: JACoW Publishing; 2022, p. 1983–6. http://dx.doi.org/10.18429/JACoW-IPAC2022-WEPOPT055.

[40] Alemany Fernandez R, Albright S, Angoletta ME, Bellodi G, Biancacci N, Bodart D, et al. Performance of the low energy ion ring at CERN with lead ions in 2022. In: Internal note. CERN; 2023, URL https://cds.cern.ch/record/2872882.

[41] Alemany-Fernandez R, Frassier A, Rey A, Wetton C, Mutin C, Gamba D, et al. Performance of the low energy ion ring at CERN with lead ions in 2022. In: Proc. IPAC'23. Geneva, Switzerland: JACoW Publishing; 2023, p. 1648–51. http://dx.doi.org/10.18429/JACoW-IPAC2023-TUPA151.

[42] Kain V, Arrutia P, Bartosik H, Follin F, King Q, Michels O, et al. Eliminating mains noise with machine learning. In: Presented at the workshop on machine learning applications for particle accelerators 2025. Geneva, Switzerland: CERN; 2025, URL https://indico.cern.ch/event/1382428/contributions/6283299/.

[43] Madysa N. Machine learning for accelerators at CERN. In: GANIL community meeting 2022. 2022, p. 1–15, URL https://gcm2022.sciencesconf.org/data/pages/gcm2022_madysa_ml_for_accelerators.pdf.

[44] Menor de Oñate A, Bruchon N, Kain V, Schenk M. Autonomous trajectory steering of DC beams at CERN's transfer lines using reinforcement learning. In: Presented at the workshop on reinforcement learning for autonomous accelerators. Hamburg, Germany: DESY; 2025, URL https://indico.kit.edu/event/4216/contributions/19231/.

[45] Kain V, Bruchon N, Hirlaender S, Küchler D, Rodriguez Mateos B, Schenk M. Continuous data-driven control of the GTS-LHC ion source at CERN. In: Proc. eCRIS'24. Geneva, Switzerland: JACoW Publishing; 2024, p. 1, URL https://indico.jacow.org/event/78/contributions/6167/.

[46] Kain V, Albright S, Alemany-Fernández R, Angoletta M, Antoniou F, Argyropoulos T, et al. Achievements and performance prospects of the upgraded LHC injectors. In: Proc. IPAC'22. Geneva, Switzerland: JACoW Publishing; 2022, p. 1610–5. http://dx.doi.org/10.18429/JACoW-IPAC2022-WEIYGD1.

[47] Trad G, et al. Esults and plans for integration of automation and optimization in operation. In: Presented at the CERN joint accelerator performance workshop. 2024, URL https://indico.cern.ch/event/1439972/contributions/6159155/.

[48] Appel S, Oeftiger A, Weick H, Madysa N, Pietri S, Kazantseva E, et al. Automated optimization of accelerator settings at GSI. In: Proc. IPAC'24. Geneva, Switzerland: JACoW Publishing; 2024, p. 882–5. http://dx.doi.org/10.18429/JACoW-IPAC2024-MOPS68.

[49] Appel S, Hirlaender S, Madysa N. Data-driven model predictive control for automated optimization of injection into the SIS18 synchrotron. In: Proc. IPAC'24. Geneva, Switzerland: JACoW Publishing; 2024, p. 1800–3. http://dx.doi.org/10.18429/JACoW-IPAC2024-TUPS59.

[50] Appel S. Machine learning and advanced accelerator optimisation at GSI/-FAIR. In: ICAP'24. 2024, p. 1–16, URL https://indico.gsi.de/event/19249/contributions/82641.

[51] Abdulrahman MM, Niu Y. Multi-objective evolutionary algorithm with decomposition for enhanced community detection in signed networks. KHWARIZMIA 2023;2023:10–23. http://dx.doi.org/10.70470/KHWARIZMIA/2023/002.

[52] Blank J, Deb K. pymoo: Multi-objective optimization in python. IEEE Access 2020;8:89497–509.