

HADRON-2030

Access to Physics with Hadrons at the
Frontier of Knowledge: Fundamental
Research and Applications

Automating Accelerator Tuning at GSI/FAIR Why we need python in the control room?

Sabrina Appel (APH)
(Team: APH, Super-FRS, CERN, TUDa, PLUS Salzburg)



TwinRISE (Trusted AI-Generated Digital Twins
for Research Infrastructures & Scientific
Excellence)



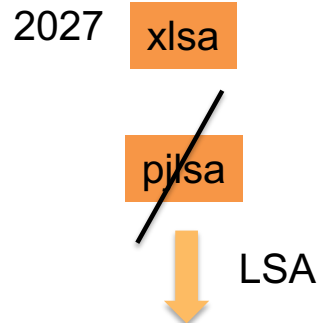
iRIS (Intelligent, Resilient &
Sustainable Research
Infrastructures)

GSI: Python Bridge

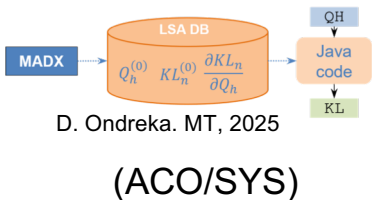


One key ingredient: Python bridge to GSI Control System (software and hardware)

-> works only at the asl cluster

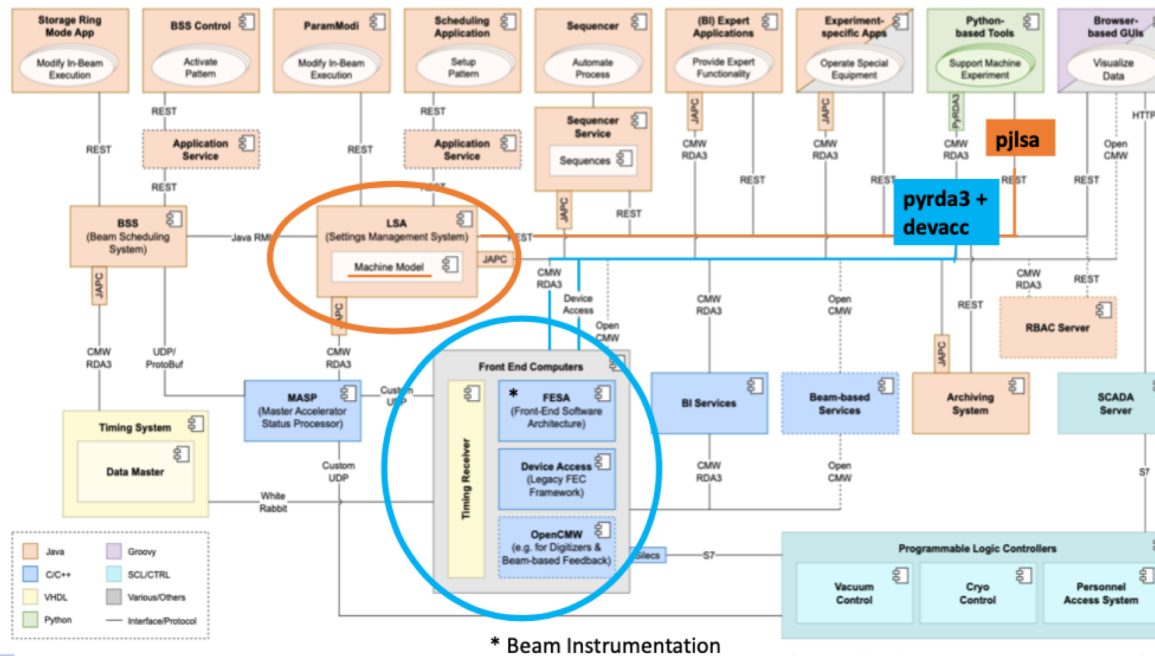


$$KL_n = KL_n^{(0)} + \frac{\partial KL_n}{\partial Q_h} \cdot (Q_h - Q_h^{(0)})$$



The Python access to **FESA** is most meaningfully used for retrieving data from **beam diagnostic devices**, while changes to **settings** (e.g., for magnets) should be handled via **LSA**.

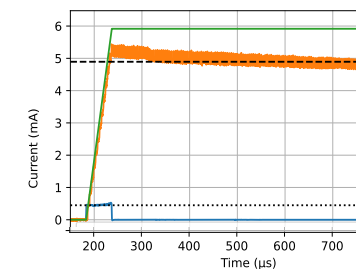
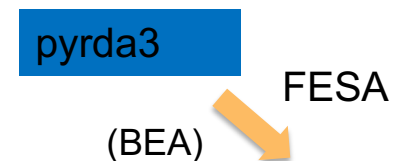
Python bridge to control system



H. Hüther. Controls Architecture. FAIR MAC Review on Controls. 2024

- ¹<https://wiki.gsi.de/BEPHY/PythonBridge>,
- ²https://git.gsi.de/scripting-tools/pilsa_gsipro,
- ³<https://www-acc.gsi.de/wiki/FESA/PythonCmwRda>,
- ⁴<https://gitlab.cern.ch/acc-co/devops/python/prototypes/pyda>

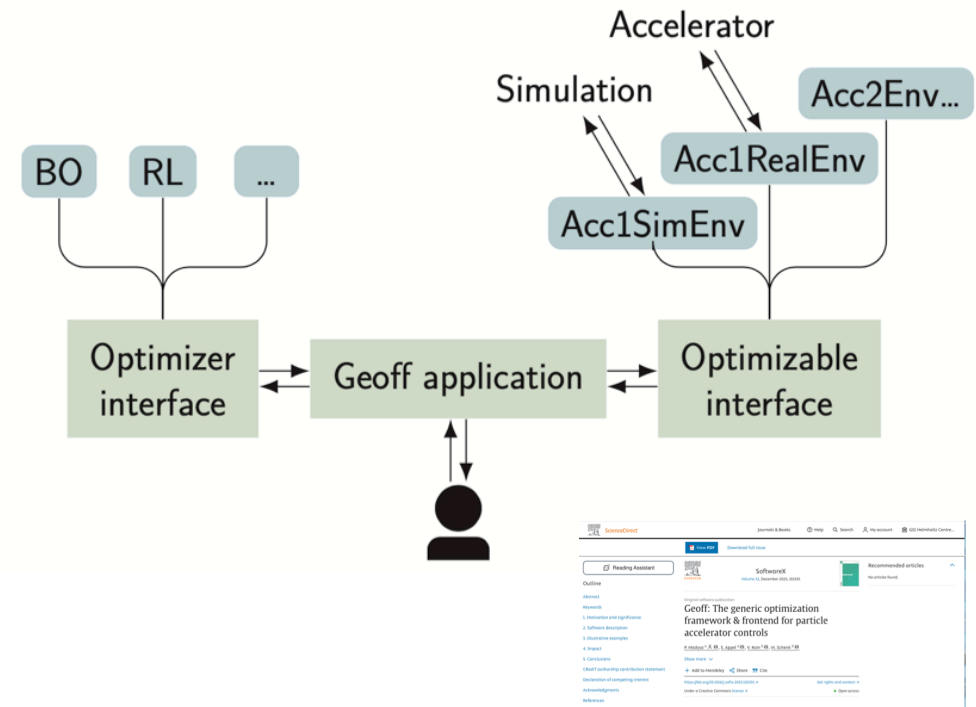
- Ported **existing** CERN solution to GSI
- Python bridge will be maintained by APH (new software engineer)



Generic Optimization Frontend & Framework (GeOFF)

Individual Python-based efforts led to **uplicated** work, **incompatibilities**, and **poor usability**.

- **Python**-based framework (modern ML: TensorFlow, PyTorch)
- lists, configures and runs optimization problems
- standardized interfaces and adapters for various packages via Common Optimization Interfaces
- **Optimization problems formulated as environment (classes)** (allow optimization of very complex problems)
- Class contains logic for live plotting, data logging, and communication with LSA, FESA + **EPCIS**
- **Quick adaptation of code and on-the-fly during shifts:** This is made easy due to flexibility of the framework.
- **Geoff GUI:** Operators don't need to juggle Python scripts



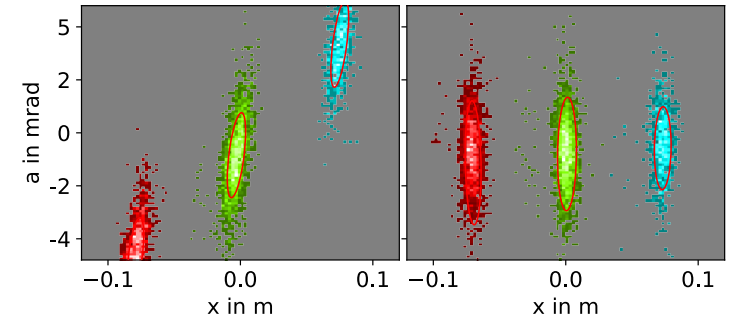
<https://doi.org/10.1016/j.softx.2025.102335>

Geoff highlights from operations

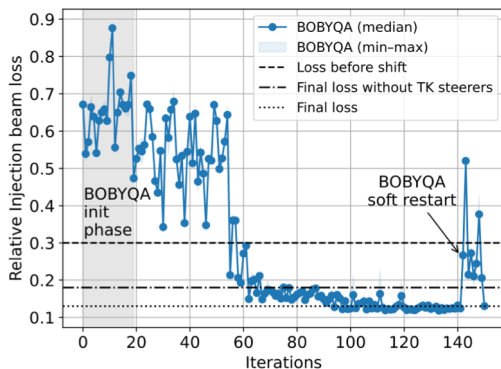
- Geoff enables rapid, automated tuning
- Open-source framework for accelerator
- Recent papers in *SoftwareX* + *EPJ Res. Infrastruct*
- Geoff will participate in several new EU grant programs (TwinRise, iRIS, and HADRON-2030)
- Collaboration with CERN, KIT + GANIL.
- **Super-FRS commissioning methods**

Separator tuning

New method ready for integral Super-FRS deployment



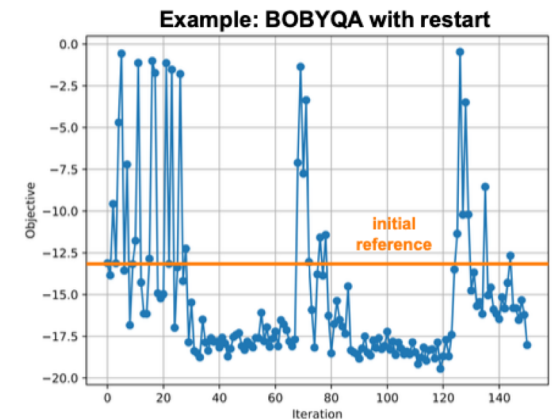
Synchrotron tuning



SIS18 injection loss reduced to ~12 %

PUMA beamline tuning

Efficiency increased from ~20 % to ~50 % in RFQcb (ion cooling and stacking)



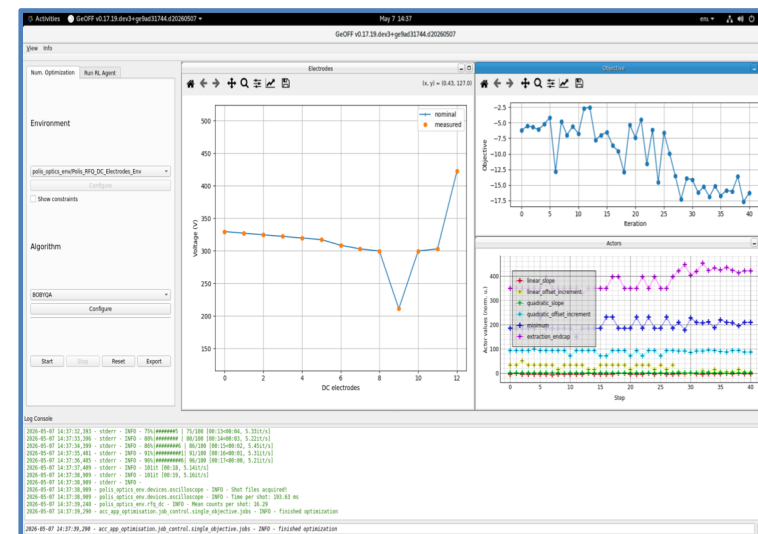
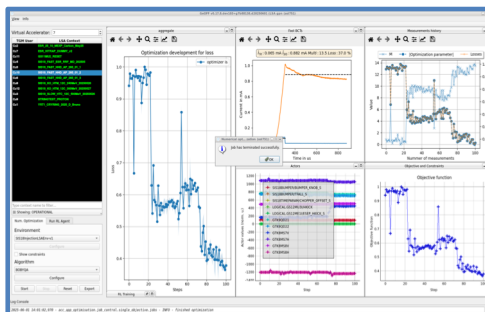
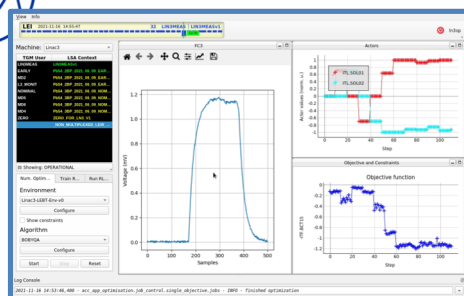
More:

S. Appel et al.: *Automating accelerator tuning at GSI/FAIR*, EPJ Res. Infrastruct. 10, 23 (2026)

Accessibility & Professional Release



Core GUI



Idea: Each facility can add its own control-system-specific features (e.g., context selector).

Control systems: **LSA (FESA)**, **legacy GSI systems**, **EPCIS**

Code hosting

Easy Installation

Documentation



<https://gitlab.cern.ch/geoff>
<https://github.com/geoff-project>

PyPI

`pip install Geoff-app`



<https://geoff.docs.cern.ch/>

Penny Madysa, Lisa Dingeldein, 2023-2026

To manage Python packages securely, CERN uses Acc-Py and rather than relying entirely on PyPI.

Introducing Python as a Supported Language for Accelerator Controls at CERN

P. Elson, C. Baldi, I. Sinkarenko, CERN, Geneva, Switzerland



Introduction

In 2019, Python was adopted as an officially supported language for interacting with CERN's accelerator controls. Highlighted here is some of the key infrastructure put in place to facilitate a user friendly, idiomatic and stable Python platform fit for operational high-level accelerator controls.

Acc-Py base distribution

- **Lightweight**, easy to install, used on local disk, containers and network drives
- Designed to be **installed side-by-side** with other versions of Acc-Py base
- Each invocation logged
- Not a pre-prepared fully-fledged distribution such as Anaconda or LCG
=> **Designed to be extended** by the user



Virtual environment

- A user can create an unlimited number of virtual environments
- No special privileges required
- Python interpreter and standard library linked from the base distribution

```
venv
├── bin
│   ├── python (symlink to base)
│   └── ...
├── lib
│   ├── pythonX.Y
│   │   └── site-packages
│   │       └── <venv-packages>
│   └── include
├── share
└── pyvenv.cfg
```

pip install

- User can choose which packages to install
- Pre-configured to use the Acc-Py package index
- CERN-specific packages combined with those from the Python Package Index (PyPI)
- Tools built on top of pip (e.g. poetry) can be used

Python well-adapted for:

- ✓ Data processing and analysis
- ✓ Machine learning & deep learning
- ✓ System automation
- ✓ Rapid prototyping
- ✓ GUI development
- ✓ Mathematical modelling
- ✓ Hardware interfacing
- ✓ ...

Examples at CERN:

- Machine development (MD) studies
- **Linac4 Source autopilot** (WEPV018)
- **Timing sequencer GUI** (THPV015)
- Expert GUI applications
- Physics simulations
- Numerical optimisation for operations
- Rapid hardware interfacing
- ...

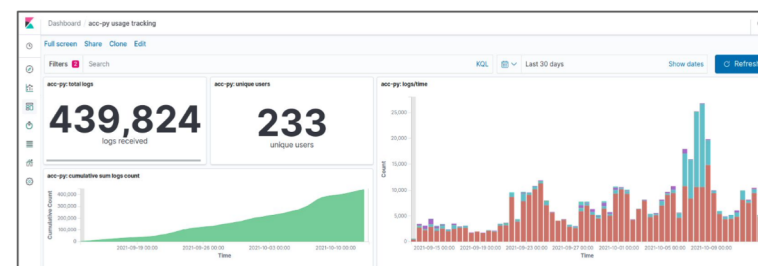
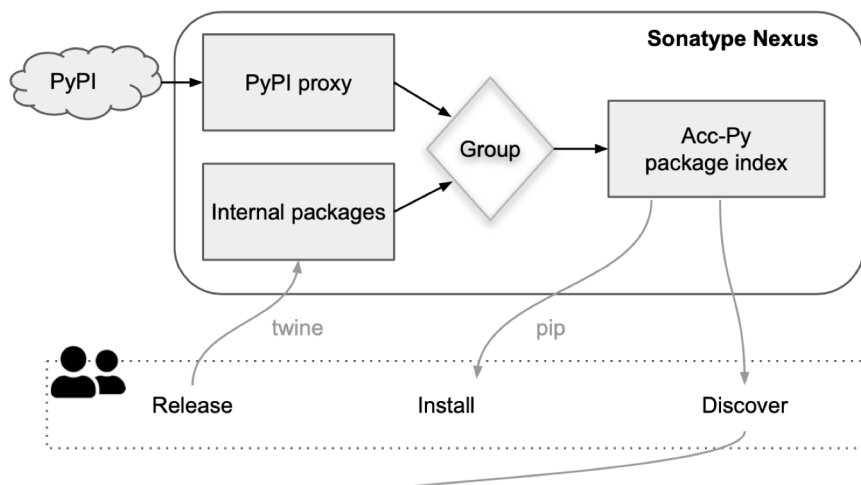
[link](#)

Introducing Python as a Supported Language for Accelerator Controls at CERN

P. Elson, C. Baldi, I. Sinkarenko, CERN, Geneva, Switzerland



Acc-Py package index



Each Acc-Py Python invocation is logged, along with data including the time, machine, username and a list of all packages installed in the environment.

This data has been invaluable in helping inform both technical and strategic decisions, and it provides a snapshot of usage of Python for accelerator controls.

In numbers, there are around **10,000** Python invocations per day and over **200** unique active users (September 2021).

2021

[link](#)

Introducing Python as a Supported Language for Accelerator Controls at CERN

P. Elson, C. Baldi, I. Sinkarenko, CERN, Geneva, Switzerland



Development and deployment tool

Step 1: Create a project

```
$ acc-py init my-project
=> Writing setup.py
=> Writing README.md
=> Configuring my_project's directory structure
```

```
./my-project/
├── my_project/
│   ├── __init__.PY
│   └── tests/
│       ├── __init__.PY
│       └── test_*.PY
├── README.md
└── setup.py
```

- Create a new Python package following a simple structure
- **Tests** are a first-class component of the project
- Possibility to generate configuration for **API Documentation** (sphinx) and **Continuous integration** (GitLab-CI)

Step 2: Freeze dependencies

```
$ acc-py app lock ./my-project/
```

```
./my-project/deployment/app
├── config.json
├── requirements-jar.json
└── requirements.txt
```

- Resolve floating dependency versions
- Including Java dependencies
- Guarantees that the **dependencies in production are the same** as those tested in development

Step 3: Deploy to central location

```
$ acc-py app deploy ./my-project/
=> Building wheel for my-project
=> Creating app in deployment location
=> my-project version 1.2.3 was deployed
```

```
/deployment/location/my-project
├── 1.2.3
│   ├── exec
│   ├── __init__
│   └── venv
├── __init__
├── app-config.yml
└── audit.log
```

- **Consistent & repeatable deployment**
- Based on a virtual environment
- **Auditable** history of application changes
- Common location for deployments

Step 4: Run in production

```
$ acc-py app run my-project
```

- Always runs in "isolated" mode
 - => **Consistent behaviour** no matter the executing user or current directory
- Runs as a Python module, no additional scripting required

[link](#)

Journal of Software: Evolution and Process

WILEY

Software: Evolution and Process

RESEARCH ARTICLE - EMPIRICAL

OPEN ACCESS

Security Assessment of Private Package Repositories: An Experience on Acc-Py at CERN

Michele Lizzit¹ | Francesco Pinzauti² | Marino Miculan¹ | Vincenzo Riccio¹

¹University of Udine, Udine, Italy | ²CERN, Meyrin, Switzerland

Correspondence: Michele Lizzit (michele.lizzit@uniud.it)

Received: 19 January 2026 | Revised: 30 June 2026 | Accepted: 17 July 2026

ABSTRACT

The introduction of package managers had a huge impact on software development, as they facilitate dependency management and the access to reusable code components. However, reliance on centralized repositories introduces security risks, as they are increasingly exploited in supply chain attacks. To reduce these risks, many companies rely on private package managers and repositories. Although public package repositories have been extensively studied, private ones remain underexplored. This paper presents a security comparison of private package repositories versus their public counterparts, through our experience on PyPI and CERN's Acc-Py. We perform a comprehensive security assessment of both repositories, complemented by discussions with the CERN development team. Using open-source static analyzers, we find that Acc-Py hosts packages with fewer potential security issues than those on the broad PyPI ecosystem. However, it remains susceptible to dependency confusion attacks due to namespace collisions with PyPI. Our dynamic analysis technique identifies telemetry collection as a privacy-monitoring trade-off. Our study provides valuable insights for analyzing and strengthening the security of private repositories, addressing their unique security challenges and attack surfaces.

10 | Conclusions and Future Work

...

Through static and metadata analysis, we demonstrated that Acc-Py has a **strong security posture**, as emerges from the comparison with PyPI. We identified potential vulnerabilities to **dependency confusion** and highlighted the importance of transparent telemetry collection practices.

...

TABLE 1 | Acc-Py vs. PyPI: dataset statistics.

Statistic	Acc-Py	PyPI top10k	PyPI 10%
Total number of packages	558	10,000	55,813
Total lines of code	931,005	113,046,417	86,636,608
Average lines of code per package	1668.47	11,304.64	1552.26
Median package size (KB)	19.9	43.0	13.3
Index accessibility	Private	Public	Public
Active users	1000	910,515	910,515

[link](#)