

GeOFF: Generic Optimization Framework and Frontend

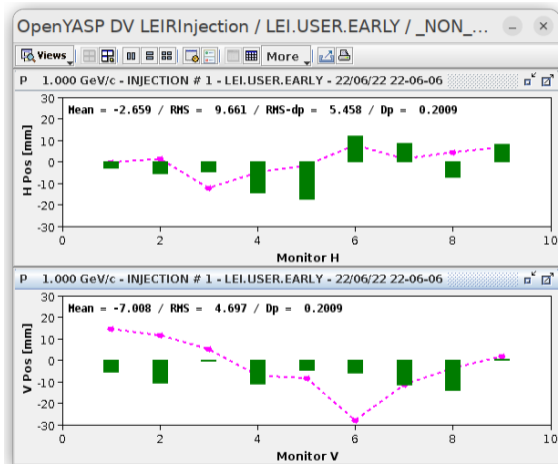
Nico Madysa

BE Seminar
1 July 2022

Motivation

many optimization problems in CERN
accelerator operations:

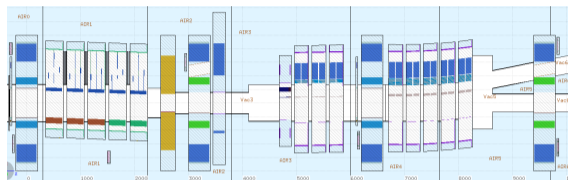
- beam steering without optics model



Motivation

many optimization problems in CERN
accelerator operations:

- beam steering without optics model
- alignment of electromagnetic septum

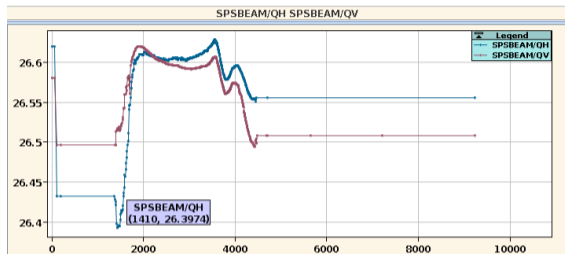


Originally developed by D. Bjorkman

Motivation

many optimization problems in CERN
accelerator operations:

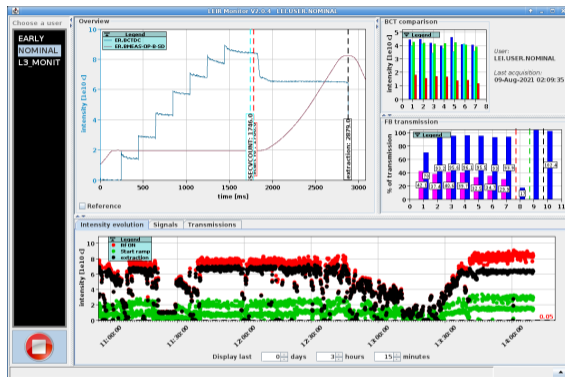
- beam steering without optics model
- alignment of electromagnetic septum
- tune optimization



Motivation

many optimization problems in CERN
accelerator operations:

- beam steering without optics model
- alignment of electromagnetic septum
- tune optimization
- simple corrector adjustments at Linac 3, ISOLDE, ...



- diverse set of machines

- diverse set of machines
- theoretical model impossible or unavailable and not worthwhile

- diverse set of machines
- theoretical model impossible or unavailable and not worthwhile
- done manually with no “obvious” algorithm to follow
(experience helps, however!)

- many different optimization algorithms exist

Motivation

- many different optimization algorithms exist
- machine learning on the rise!

Motivation

- many different optimization algorithms exist
- machine learning on the rise!
- each package has slightly different API

- many different optimization algorithms exist
- machine learning on the rise!
- each package has slightly different API
- most algorithms take control, don't account for pauses, disturbances, cancellations, replays

LEIR Steering

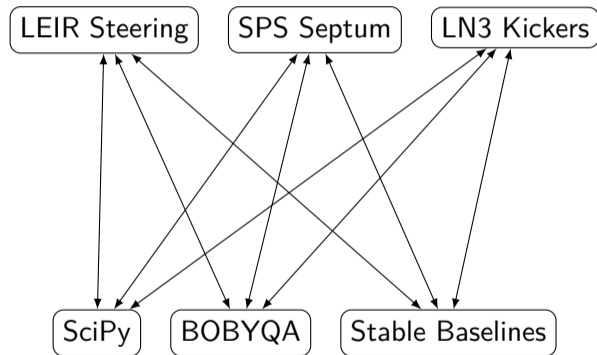
SPS Septum

LN3 Kickers

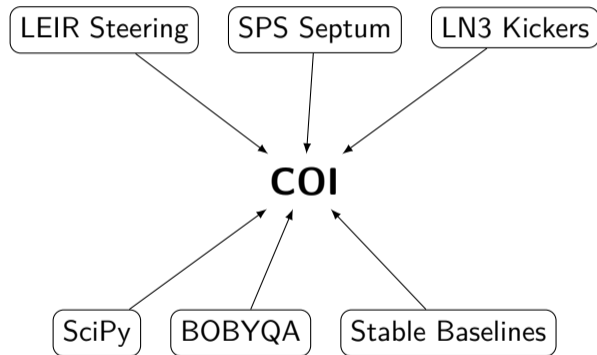
- many different optimization problems

Motivation

- many different optimization problems
- many different optimizers

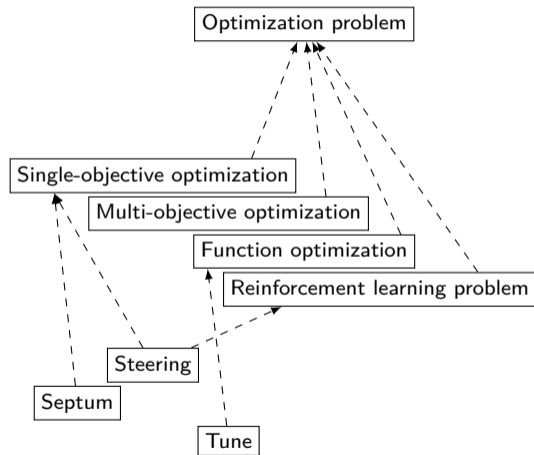


- many different optimization problems
- many different optimizers
- avoid combinatorial explosion with **common optimization interfaces**



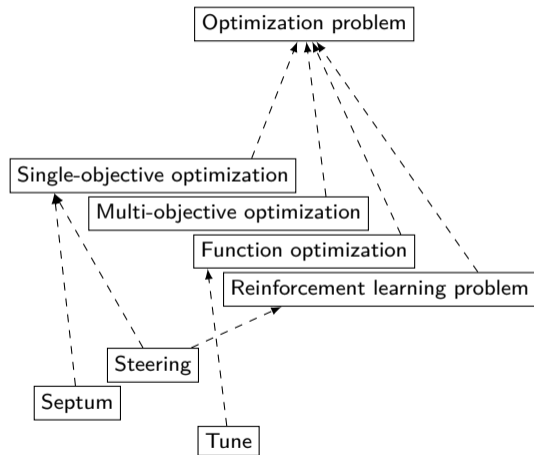
Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages



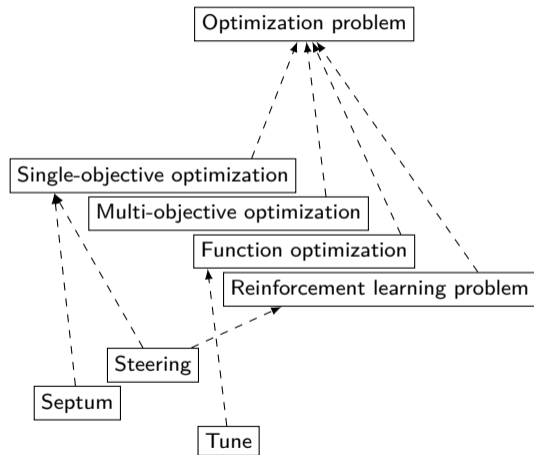
Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- “20 % programming, 80 % documentation”



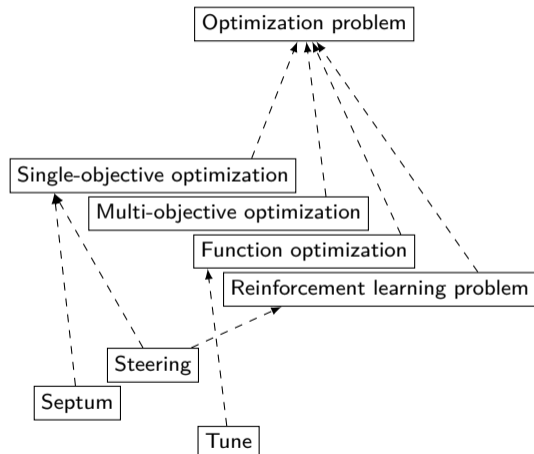
Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- “20 % programming, 80 % documentation”
- inspired by **OpenAI Gym**



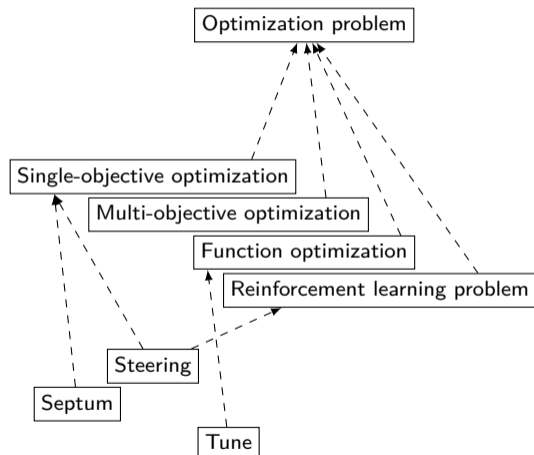
Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- “20 % programming, 80 % documentation”
- inspired by **OpenAI Gym**
- extends their interfaces to numeric optimization



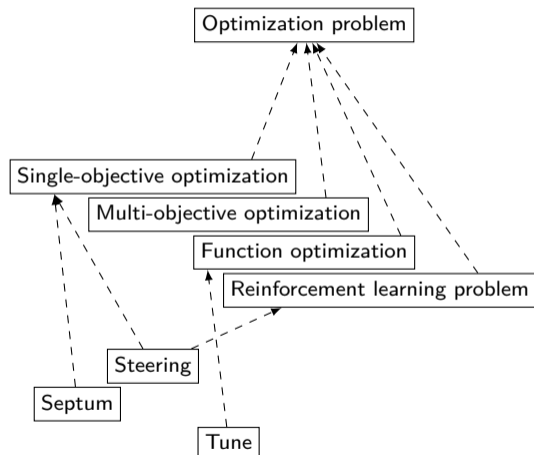
Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- “20 % programming, 80 % documentation”
- inspired by **OpenAI Gym**
- extends their interfaces to numeric optimization
- extend Gym metadata system with CERN-specific info



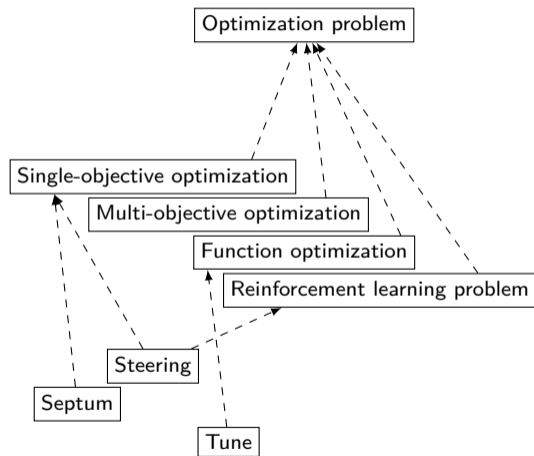
Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- “20 % programming, 80 % documentation”
- inspired by **OpenAI Gym**
- extends their interfaces to numeric optimization
- extend Gym metadata system with CERN-specific info
 - ▶ which accelerator?



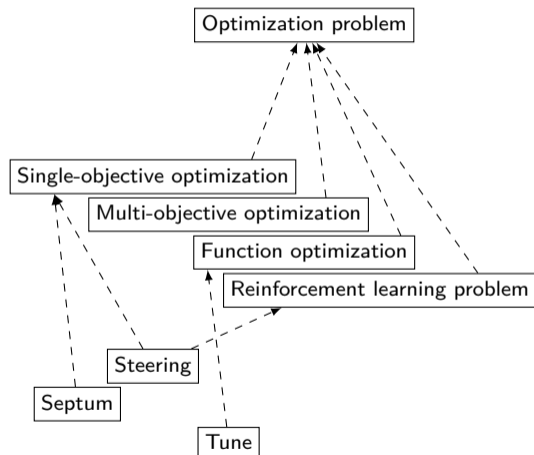
Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- “20 % programming, 80 % documentation”
- inspired by **OpenAI Gym**
- extends their interfaces to numeric optimization
- extend Gym metadata system with CERN-specific info
 - ▶ which accelerator?
 - ▶ communicates with machines?



Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- “20 % programming, 80 % documentation”
- inspired by **OpenAI Gym**
- extends their interfaces to numeric optimization
- extend Gym metadata system with CERN-specific info
 - ▶ which accelerator?
 - ▶ communicates with machines?
 - ▶ wants to plot additional data?



Common Optimization Interfaces (COI)

- extensively documented at <https://cernml-coi.docs.cern.ch/>

The screenshot shows the documentation page for Common Optimization Interfaces (COI) from the CERN ML project. The page title is "Common Optimization Interfaces" and it is part of the "cernml-coi 0.8.4.dev4+g9fddee0 documentation". The page content includes an introduction to the project, a description of the COI interfaces, and a list of additional features provided by the "cernml-coi-utils" package. A sidebar on the right contains navigation links for "Next topic", "Tutorials", "This Page", "Show Source", and "Quick search".

cernml-coi 0.8.4.dev4+g9fddee0 documentation » next | modules | index

Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex.

CERNML-COI defines common interfaces that facilitate using numerical optimization and reinforcement learning (RL) on the same optimization problems. This makes it possible to unify both approaches into a generic optimization application in the CERN Control Center.

The [cernml-coi-utils](#) package provides many additional features that complement the COIs.

This repository can be found online on CERN's [Gitlab](#).

- [Tutorials](#)
 - [Packaging Crash Course](#)
 - [Implementing SingleOptimizable](#)
- [User Guide](#)
 - [The Core API](#)
 - [Problem Registry](#)
 - [Synchronization and Cancellation](#)
 - [Other Interfaces](#)
 - [Optimization of LSA Functions](#)
- [API Reference](#)
 - [Common Optimization Interfaces](#)
 - [Spaces](#)
 - [Configuration of Problems](#)
 - [Problem Registry](#)
 - [Separable and Goal-Based Interfaces](#)
 - [Problem Checkers](#)
 - [Cancellation](#)
- [Changelog](#)
 - [Unreleased](#)
 - [v0.8.3](#)

Next topic

Tutorials

This Page

Show Source

Quick search

Go

Common Optimization Interfaces (COI)

- extensively documented at <https://cernml-coi.docs.cern.ch/>
- describes expected and allowed behavior for each interface

cernml-coi 0.8.4.dev4+g9fddee0 documentation » next | modules | index

Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex.

CERNML-COI defines common interfaces that facilitate using numerical optimization and reinforcement learning (RL) on the same optimization problems. This makes it possible to unify both approaches into a generic optimization application in the CERN Control Center.

The [cernml-coi-utils](#) package provides many additional features that complement the COIs.

This repository can be found online on CERN's [Gitlab](#).

- [Tutorials](#)
 - [Packaging Crash Course](#)
 - [Implementing SingleOptimizable](#)
- [User Guide](#)
 - [The Core API](#)
 - [Problem Registry](#)
 - [Synchronization and Cancellation](#)
 - [Other Interfaces](#)
 - [Optimization of LSA Functions](#)
- [API Reference](#)
 - [Common Optimization Interfaces](#)
 - [Spaces](#)
 - [Configuration of Problems](#)
 - [Problem Registry](#)
 - [Separable and Goal-Based Interfaces](#)
 - [Problem Checkers](#)
 - [Cancellation](#)
- [Changelog](#)
 - [Unreleased](#)
 - [v0.8.3](#)

Common Optimization Interfaces (COI)

- extensively documented at <https://cernml-coi.docs.cern.ch/>
- describes expected and allowed behavior for each interface
- includes several (2) tutorials

cernml-coi 0.8.4.dev4+g9fddee0 documentation » next | modules | index

Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex.

CERNML-COI defines common interfaces that facilitate using numerical optimization and reinforcement learning (RL) on the same optimization problems. This makes it possible to unify both approaches into a generic optimization application in the CERN Control Center.

The [cernml-coi-utils](#) package provides many additional features that complement the COIs.

This repository can be found online on CERN's [Gitlab](#).

- [Tutorials](#)
 - [Packaging Crash Course](#)
 - [Implementing SingleOptimizable](#)
- [User Guide](#)
 - [The Core API](#)
 - [Problem Registry](#)
 - [Synchronization and Cancellation](#)
 - [Other Interfaces](#)
 - [Optimization of LSA Functions](#)
- [API Reference](#)
 - [Common Optimization Interfaces](#)
 - [Spaces](#)
 - [Configuration of Problems](#)
 - [Problem Registry](#)
 - [Separable and Goal-Based Interfaces](#)
 - [Problem Checkers](#)
 - [Cancellation](#)
- [Changelog](#)
 - [Unreleased](#)
 - [v0.8.3](#)

Next topic

Tutorials

This Page

Show Source

Quick search Go

Common Optimization Interfaces (COI)

- extensively documented at <https://cernml-coi.docs.cern.ch/>
- describes expected and allowed behavior for each interface
- includes several (2) tutorials
- both surface-level **user guides** and in-depth **API references**

The screenshot shows the documentation page for Common Optimization Interfaces (COI) on the CERN ML project website. The page title is "Common Optimization Interfaces" and it includes a navigation bar with "next", "modules", and "index" links. The main content area starts with a section header "Common Optimization Interfaces" followed by a paragraph explaining that CERN ML is a project for bringing numerical optimization, machine learning, and reinforcement learning to the CERN accelerator complex. It then defines the COI interfaces and mentions the "cernml-coi-utils" package. A sidebar on the right contains navigation buttons for "Next topic", "Tutorials", "This Page", "Show Source", and "Quick search". The main content area lists several topics under "Tutorials", "User Guide", "API Reference", and "Changelog".

cernml-coi 0.8.4.dev4+g9fddee0 documentation » next | modules | index

Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex.

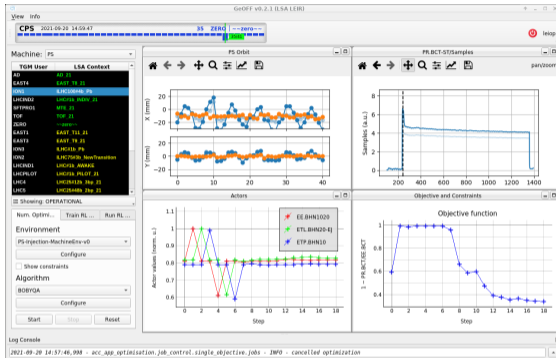
CERNML-COI defines common interfaces that facilitate using numerical optimization and reinforcement learning (RL) on the same optimization problems. This makes it possible to unify both approaches into a generic optimization application in the CERN Control Center.

The [cernml-coi-utils](#) package provides many additional features that complement the COIs.

This repository can be found online on CERN's [Gitlab](#).

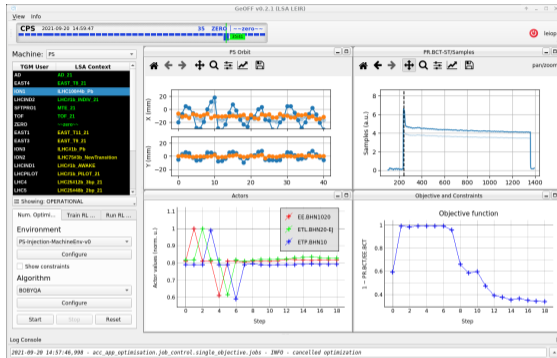
- [Tutorials](#)
 - [Packaging Crash Course](#)
 - [Implementing SingleOptimizable](#)
- [User Guide](#)
 - [The Core API](#)
 - [Problem Registry](#)
 - [Synchronization and Cancellation](#)
 - [Other Interfaces](#)
 - [Optimization of LSA Functions](#)
- [API Reference](#)
 - [Common Optimization Interfaces](#)
 - [Spaces](#)
 - [Configuration of Problems](#)
 - [Problem Registry](#)
 - [Separable and Goal-Based Interfaces](#)
 - [Problem Checkers](#)
 - [Cancellation](#)
- [Changelog](#)
 - [Unreleased](#)
 - [v0.8.3](#)

Generic Optimization Frontend & Framework (GeOFF)



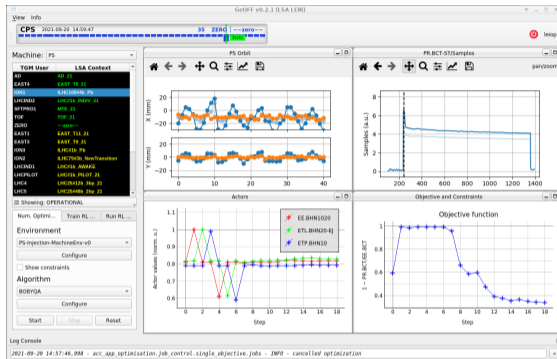
- GUI app based on PyQt5 and **AccWidgets** (BE-developed CERN-specific GUI widgets)

Generic Optimization Frontend & Framework (GeOFF)



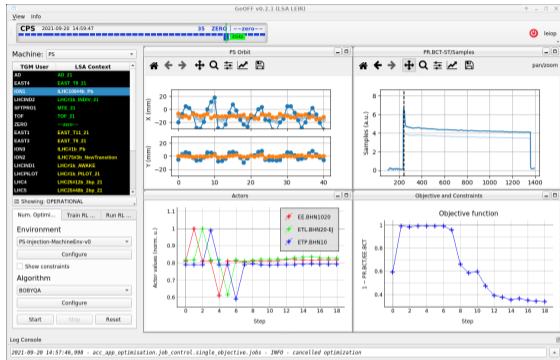
- GUI app based on PyQt5 and **AccWidgets** (BE-developed CERN-specific GUI widgets)
- lists, configures and runs optimization problems

Generic Optimization Frontend & Framework (GeOFF)



- GUI app based on PyQt5 and **AccWidgets** (BE-developed CERN-specific GUI widgets)
- lists, configures and runs optimization problems
- very extensible: optimization problems are loaded as plugins

Generic Optimization Frontend & Framework (GeOFF)



- GUI app based on PyQt5 and **AccWidgets** (BE-developed CERN-specific GUI widgets)
- lists, configures and runs optimization problems
- very extensible: optimization problems are loaded as plugins
- built-in list of optimizers plus runtime load mechanism

- 1 implement class that inherits from COI (~ 300 lines of code)

- 1 implement class that inherits from COI (~ 300 lines of code)
 - ▶ declare metadata

- 1 implement class that inherits from COI (~ 300 lines of code)
 - ▶ declare metadata
 - ▶ implement machine communication

- 1 implement class that inherits from COI (~ 300 lines of code)
 - ▶ declare metadata
 - ▶ implement machine communication
 - ▶ add plotting (if any)

- 1 implement class that inherits from COI (~ 300 lines of code)
 - ▶ declare metadata
 - ▶ implement machine communication
 - ▶ add plotting (if any)
- 2 dynamically load package into GeOFF

- 1 implement class that inherits from COI (~ 300 lines of code)
 - ▶ declare metadata
 - ▶ implement machine communication
 - ▶ add plotting (if any)
- 2 dynamically load package into GeOFF

Optionally:

- 3 upload code to CERN package index (provided by **our Python team**)

- 1 implement class that inherits from COI (~ 300 lines of code)
 - ▶ declare metadata
 - ▶ implement machine communication
 - ▶ add plotting (if any)
- 2 dynamically load package into GeOFF

Optionally:

- 3 upload code to CERN package index (provided by **our Python team**)
- 4 notify us to get package integrated into GeOFF

Use cases

- PS:

Use cases

- PS:
 - ▶ steering from PS to n-TOF

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection
 - ▶ transfer line to PS

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection
 - ▶ transfer line to PS
- SPS:

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection
 - ▶ transfer line to PS
- SPS:
 - ▶ ZS alignment

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection
 - ▶ transfer line to PS
- SPS:
 - ▶ ZS alignment
 - ▶ crystal shadowing

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection
 - ▶ transfer line to PS
- SPS:
 - ▶ ZS alignment
 - ▶ crystal shadowing
 - ▶ tune adjustments to avoid resonances

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection
 - ▶ transfer line to PS
- SPS:
 - ▶ ZS alignment
 - ▶ crystal shadowing
 - ▶ tune adjustments to avoid resonances
 - ▶ longitudinal blowup (superseded by theoretical model)

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection
 - ▶ transfer line to PS
- SPS:
 - ▶ ZS alignment
 - ▶ crystal shadowing
 - ▶ tune adjustments to avoid resonances
 - ▶ longitudinal blowup (superseded by theoretical model)
 - ▶ splitter loss optimization between North Area experiments

Use cases

- PS:
 - ▶ steering from PS to n-TOF
 - ▶ resonance compensation (also in PSB)
- Linac3: LEBT steering
- LEIR:
 - ▶ transfer line from Linac3 and injection
 - ▶ multi-turn injection
 - ▶ transfer line to PS
- SPS:
 - ▶ ZS alignment
 - ▶ crystal shadowing
 - ▶ tune adjustments to avoid resonances
 - ▶ longitudinal blowup (superseded by theoretical model)
 - ▶ splitter loss optimization between North Area experiments
- ISOLDE: generic transfer line optimizer under investigation

Next steps

- turn GUI into modular components

Next steps

- turn GUI into modular components
 - ▶ provide each part *and* the whole GUI as a library

Next steps

- turn GUI into modular components
 - ▶ provide each part *and* the whole GUI as a library
 - ▶ experiments & machines can package GUI with their own plugins

Next steps

- turn GUI into modular components
 - ▶ provide each part *and* the whole GUI as a library
 - ▶ experiments & machines can package GUI with their own plugins
 - ▶ gives control to the users, avoids bottleneck in release process

Next steps

- turn GUI into modular components
 - ▶ provide each part *and* the whole GUI as a library
 - ▶ experiments & machines can package GUI with their own plugins
 - ▶ gives control to the users, avoids bottleneck in release process
- data recording, automatic model training

Next steps

- turn GUI into modular components
 - ▶ provide each part *and* the whole GUI as a library
 - ▶ experiments & machines can package GUI with their own plugins
 - ▶ gives control to the users, avoids bottleneck in release process
- data recording, automatic model training
- multi-objective optimization

Next steps

- turn GUI into modular components
 - ▶ provide each part *and* the whole GUI as a library
 - ▶ experiments & machines can package GUI with their own plugins
 - ▶ gives control to the users, avoids bottleneck in release process
- data recording, automatic model training
- multi-objective optimization
- integration with Machine Learning Platform

Next steps

- turn GUI into modular components
 - ▶ provide each part *and* the whole GUI as a library
 - ▶ experiments & machines can package GUI with their own plugins
 - ▶ gives control to the users, avoids bottleneck in release process
- data recording, automatic model training
- multi-objective optimization
- integration with Machine Learning Platform
 - ▶ apply stand-alone models to given optimization problem

Next steps

- turn GUI into modular components
 - ▶ provide each part *and* the whole GUI as a library
 - ▶ experiments & machines can package GUI with their own plugins
 - ▶ gives control to the users, avoids bottleneck in release process
- data recording, automatic model training
- multi-objective optimization
- integration with Machine Learning Platform
 - ▶ apply stand-alone models to given optimization problem
 - ▶ upload trained models

- many optimization problems and many algorithms to choose from

Summary

- many optimization problems and many algorithms to choose from
- we present a standardized interface to connect the two

Summary

- many optimization problems and many algorithms to choose from
- we present a standardized interface to connect the two
- extensive documentation with guides, examples and in-depth references

Summary

- many optimization problems and many algorithms to choose from
- we present a standardized interface to connect the two
- extensive documentation with guides, examples and in-depth references
- prototype GUI to present this platform to the user

Summary

- many optimization problems and many algorithms to choose from
- we present a standardized interface to connect the two
- extensive documentation with guides, examples and in-depth references
- prototype GUI to present this platform to the user
- strong focus on customizability and plugin mechanisms

Summary

- many optimization problems and many algorithms to choose from
- we present a standardized interface to connect the two
- extensive documentation with guides, examples and in-depth references
- prototype GUI to present this platform to the user
- strong focus on customizability and plugin mechanisms
- successful use at several machines