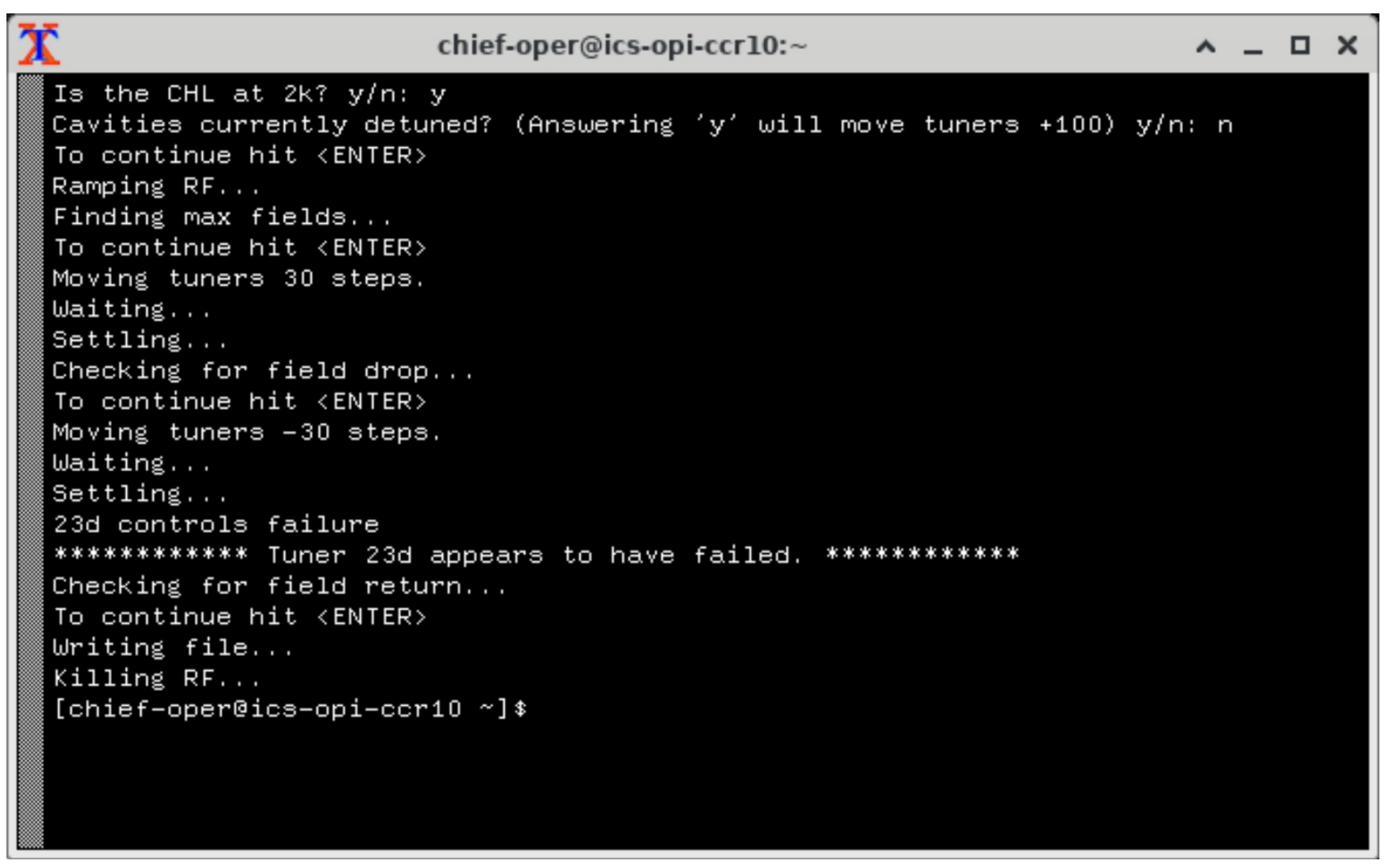
Operator-made Tools and Interfaces at SNS

J. Rye, C. Elliott, J. Mammosser, A. Zhukov - Spallation Neutron Source Oak Ridge, TN U.S.A.

Abstract

At the Spallation Neutron Source, operators have developed software for assessing tuner motor functionality on the superconducting linac (SCL). Additionally, a graphical user interface (GUI)-based application has been created to automate the optimization of beam losses and reduction of residual activation levels within the accelerator tunnels. Python and Qt made this possible. Modern high-level programming languages like Python and GUI frameworks such as Qt and Qt Designer enable individuals to quickly advance from beginners to creating sophisticated interactive applications. These tools have significantly enhanced our capacity to develop user-friendly solutions to accelerate the completion of otherwise lengthy and complex processes.

Testing SCL Tuner Motor Functionality



SCL tuners need to be tested before and after every run cycle. This process used to take on average 2 hours with the 81 cavities originally installed and could have taken 3 hours or more by the end of the Proton Power Upgrade (32 additional cavities). With the implementation of this software, we have gotten that down to just a few minutes regardless of the number of tuners to be tested.

General algorithm:

- Ramp cavities to a low gradient
- Move klystrons in 0.5 kHz increments to roughly find resonance
- Move klystrons in 0.1 kHz increments to find max fields
- Move tuners off resonance, verify field drop
- Move tuners back on resonance, verify field return

Considerations:

- Temperature of the cavity
- On resonance vs detuned at the start
- Where the tuners should end up, detuned vs on resonance
- As always, how to store the data for easy reference later

Future plans:

- Re-write in PySide6 utilizing an OOP approach
- Launching in separate threads
- Graphical User Interface

Parameters measured:

- Cavity field
- Klystron frequency
- Motor position
- Motor movement command
- Motor hard stop limit switches

Checks performed:

- Field drop on detune
- Field return on retune
- Motor position within an acceptable range of the goal after every move
- No hard limit reached
- Move and stop commands given / rescinded appropriately

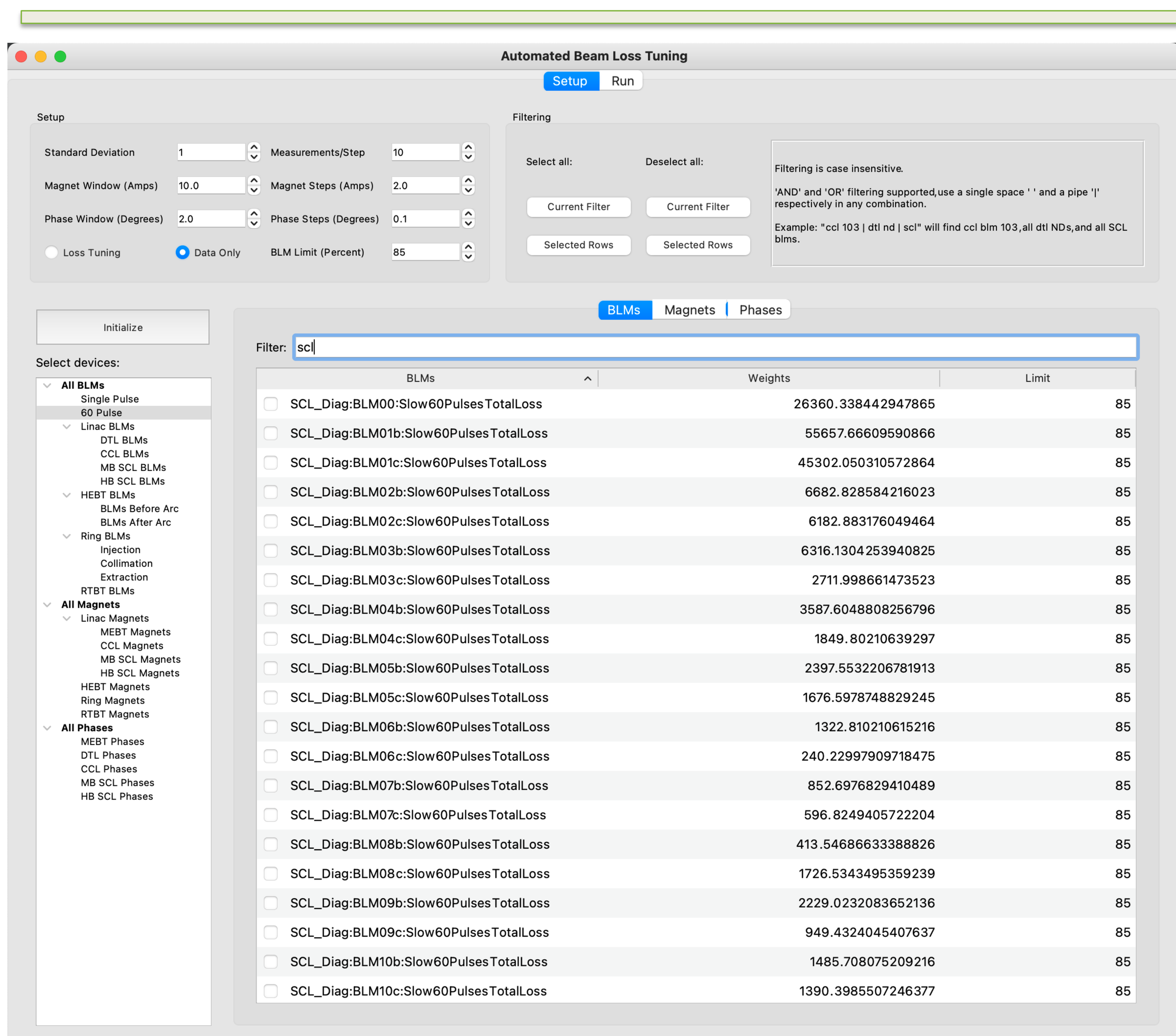
Automated Beam Loss Tuning (ABLT) Interface

The Automated Beam Loss Tuning project is a collaboration between Operations and Accelerator Physics groups with the ultimate goal of utilizing machine learning (ML) in tuning beam losses and lowering residual activation levels in the accelerator tunnels.

Operators perform the majority of the work, writing all code related to the scan, the GUI, and post-run correlation data analysis / visualization. We also perform the scans and take all the data. A member of the Accelerator Physics group, Alexander Zhukov, is our point of contact / mentor for all Python-related questions and has been invaluable in this process. Physics group will also be handling the ML side of things once we have optimized our software and taken enough data.

With development of the ABLT routine underway, authored by Carrie Elliott, we needed an interface to establish the parameters of the scan. Initially, however, the various configurations in which operators may want to perform the scan were not yet clearly defined. As a result, there was a need for a GUI that could adapt flexibly and be readily modified as our understanding of the necessary conditions evolved.

To address these considerations, the choice fell upon PyQt5 as the preferred framework. Its modular nature and ease of editing perfectly aligned with our evolving needs, allowing us to not only incorporate the required elements as they became apparent but also refine the presentation to best suit our workflow and preferences.



Setup

Parameters:

- Window size / area to be scanned
- Maximum safe step size
- Standard deviation
- Soft BLM limit, hard limit determined dynamically
- Loss tuning vs Data only

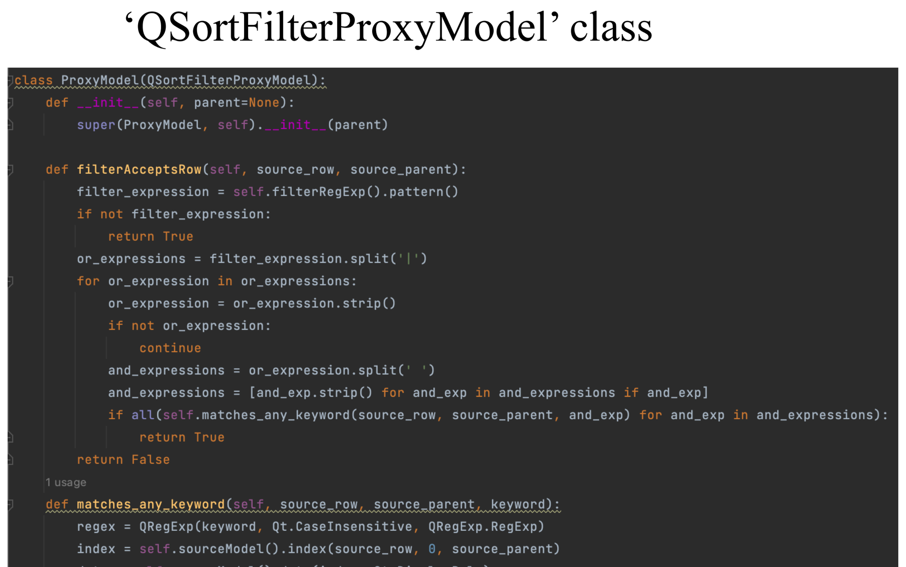
Selectable devices:

- 60 pulse BLM signals
- Single pulse BLM signals
- Quadrupole magnets
- MEBT – SCL phases

Clicking 'Initialize' loads the scan parameters as well as auto switches to the Run tab.

Filtering:

- Selecting a group from the tree will auto filter by that group
- Text input field:
  - Applied on top of tree filter, if selected
  - Single space = logical 'and'
  - Pipe character = logical 'or'
  - Custom filtering supported via override of the 'filterAcceptsRow' method of the 'QSortFilterProxyModel' class

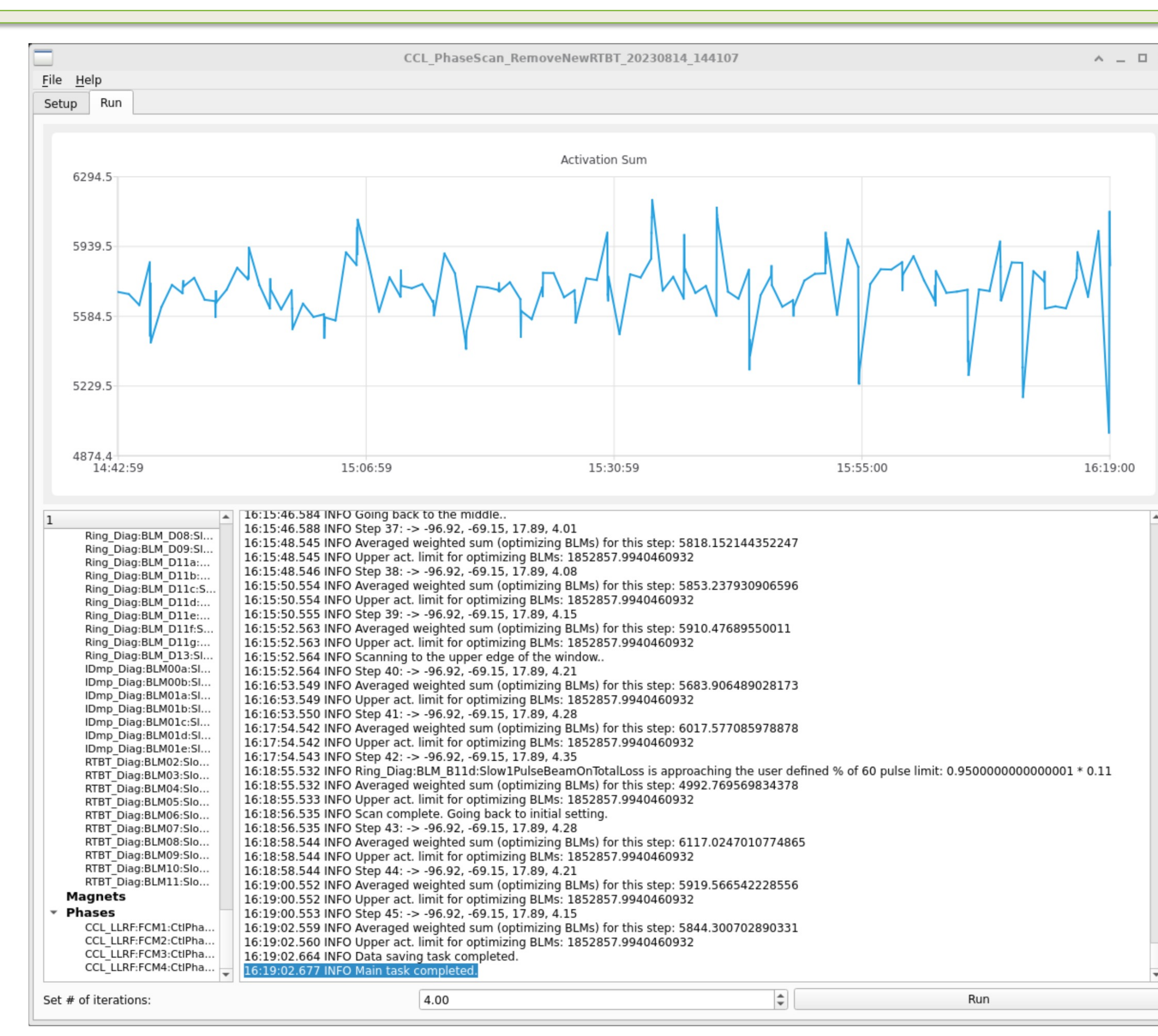


Safety features:

- Minimum / maximum window and step sizes hard coded, GUI will not accept values outside of these ranges
- Setup combo boxes will only change their respective values if the appropriate tab is selected and device rows are highlighted
- Locked to a single instance on a specific operator interface (OPI) to prevent multiple scans running simultaneously

Convenience features:

- Save / Load configurations, date and time added to filename automatically
- Curated groups of devices selectable from the tree, does not auto check however, must hit 'Select current filter' button
- Device tabs auto switch when selecting device groups from the tree
- Next scan can be configured without impacting current scan



Run

- Live plot of the calculated activation sum shows entire history of scan vs just a window of time
- Informational statements, warnings, and other information printed to the text browser
- Multiple iterations of the same scan can be performed if desired. Once the previous thread closes, another instance will be created with the same scan configuration.
- Click 'Run' once ready. Stop button not shown here but featured in a later version.

Future plans:

- Pause button
- Rewrite in PySide6, we have collectively decided to start developing in PySide vs PyQt
- Move application to a softIOC, this should make it more reliable overall as well as prevent the need to lock it to a single instance via software
- Real time data visualization, currently all done post scan
- General aesthetic improvements, suggestions welcome!
- Gain permission to tune losses during production
- Ultimately, incorporate machine learning

*** For detailed information on the scan routine, please consider attending Carrie Elliott's presentation "Development of an Automated Beam Loss Tuning Application in a High-Power Accelerator" on Thursday, Sept. 14th.

Lessons learned:

- People WILL hit the 'run' button more than once, especially if a scan is in progress. Disable it until the scan is complete.
- Just because you CAN build it, does not necessarily mean you SHOULD build it, sometimes less is more
- Python wrappers for Qt are awesome

Thanks for reading!