ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUMBCM012

# MULTI-DIMENSIONAL SPECTROGRAM APPLICATION FOR LIVE VISUALIZATION AND MANIPULATION OF LARGE WAVEFORMS

B. E. Bolling*, A. A. Gorzawski, J. Petersson, European Spallation Source ERIC, Lund, Sweden

*Abstract*

The European Spallation Source (ESS) is a research facility under construction aiming to be the world's most powerful pulsed neutron source. It is powered by a complex particle accelerator designed to provide a 2.86 ms long proton pulse at 2 GeV with a repetition rate of 14 Hz. Protons are accelerated via cavity fields through various accelerating structures that are powered by Radio Frequency (RF) power. As the cavity fields may break down due to various reasons, usually post-mortem data of such events contain the information needed regarding the cause. In other events, the underlying cause may have been visible on previous beam pulses before the interlock triggering event.

The Multi-Dimensional Spectrogram Application is designed to be able to collect, manipulate and visualize large waveforms at high repetition rates, with the ESS goal being 14 Hz, for example cavity fields, showing otherwise unnoticed temporary breakdowns that may explain the sometimes-unknown reason for increased power (compensating for those invisible temporary breakdowns). The first physical event that was recorded with the tool was quenching of a superconducting RF cavity in real time in 3D. This paper describes the application developed using Python and the pure-python graphics and GUI library PyQtGraph and PyQt5 with Python-OpenGL bindings.

## INTRODUCTION

The ability tovisualize data arrays as a spectrum on computers has developed as computer processing power increased, with many spectrograph software developed both by industry and research institutes. [1]

The first concept of this application was realized by developing a realtime spectrum visualization with a 2D heat map using Python [2] with the pyqtgraph library [3], combined with some other graphical elements via the PyQt5 Python library [4]. To integrate this with a proper handling of data streams (scalars and arrays with timestamps) as well as having the ability to import archived data, the pychiver library was used (see its subsection under Methodology). In-situ methods to manipulate were added via the standard Python NumPy library to e.g. apply a Discrete Fourier Transform on live data arrays, as well as custom-tailored functions within this package.

This paper will focus on the 3D functionality of the application, as the 3D functionalities seems in some cases that will be discussed in this paper to be more useful for visualizing live-streamed data-arrays in comparison with standard 2D waveform plots.

* benjamin.bolling@ess.eu

### *Quench of a SRF Cavity*

The idea to begin using a 3D spectrum visualization came whilst doing conditioning of Superconducting Radio Frequency (SRF) cavities and observing the so-called quench phenomena. This occurs in SRF cavities when localized heating exceeds the critical temperature or when the critical superconducting magnetic field is exceeded, which causes the cavity material to lose its superconductivity. With the final day of an SRF cavity's cold conditioning approaching, the initial prototype to be able to visualize a quench in 3D was quickly and successfully implemented. With it, a provoked quench was captured as can be seen in Fig. 1.

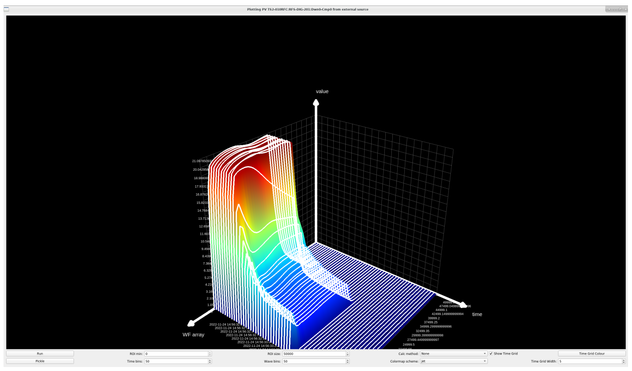

Figure 1: SRF quench captured with the first prototype of the application.

Since then, the application has been further developed to increase its reliability and functionality.

## METHODOLOGY

Multiple Python libraries are used to span the Spectrogram application, which can be split into 3 main parts:

- Data retrieval (pychiver)
- GUI (PyQt5)
- Data visualization (pyqtgraph and openGL)

### *EPICS*

Experimental Physics and Industrial Control System (EPICS) is a set of software tools and applications that provide a software infrastructure that can be used to build distributed control systems to operate components in large-scale scientific experiments, such as particle accelerators or telescopes [5].

At ESS, physical components are controlled via EPICS Input-/Output-Controllers (IOCs) that communicate with physical hardware controllers (e.g. programmable logics controllers, PLCs). Each property of the physical component (e.g. applied current or detected pressure levels) has its own so-called process variable, which is processed within the IOC and sent either to a client (e.g. a control room workstation reading or controlling the value) or with a new set-value to the hardware controlling the component. This includes both scalar values (such as a numerical, string or boolean) and data arrays (also referred to as waveforms).

In order to create Python-based software interfacing with the EPICS for real-time data visualization, the pyEpics package [6] is used.

### *Pychiver*

Independently to the use cases highlighted by this paper, a broad use python package *pychiver* [7] is available to interact with EPICS-based services at ESS. It focuses on accessing data stored in EPICS Archiver, data available in real-time, and configurations saved in the Save and Restore. One of the features used and described hereafter in the application is the waveform collector. This particular function allows for the creation of dedicated threads to collect or retrieve the data. An important function of a visualization application is the data size. For the retrieval of the archived data, a *pychiver*-native function ensures that big blocks of retrievals are avoided. In the case of real-time data, this thread comes with an internal circular buffer that allows for limiting the overall size of collected waveforms. Any pychiver waveform collector comes with the internally computed average values inside the configurable region of interest (ROI), albeit a ROI trend over all collected data is also available. All these values, in case of real-time data collection, are updated and recomputed upon the new data arrival.

### *Visualization Elements*

The main graphical user interface for the package utilizes the PyQt5 library, with pyqtgraph library components used for visualization of the data from pychiver and with python-opengl bindings [8] to render the 3D graphics. To manipulate the data arrays, standard NumPy array operators are used on the processed waveform The standard configuration of the 3D spectrogram launches with the intensity displayed by the vertical axis, the waveform position displayed by the left horizontal axis (WF (waveform) array) and the right horizontal axis showing the time the waveform array was captured.

### *Combined*

Either an event-driven data stream is set up via a connection to a PV or data is fetched from an archiver-instance. For both cases, the data is passed through a pre-processing block (restructuring incoming data to allow it to be manipulated and ensuring correct number of arrays are passed for live data), a data manipulation block (see *Live manipulation* bullet point under *Features*), followed by setting up the data correctly for the visualization. The final step is to apply the processed data to the 3D model, which is done immediately for the archived data and also for the live data if the visualizer is not paused. Hence, live data can be collected in the background whilst the user may investigate a single event in the visualizer, resulting in that the connection to a PV is not interrupted for as long as the application is running and connection permits this. A high-level flowchart depicting this is attached beneath in Fig. 2.

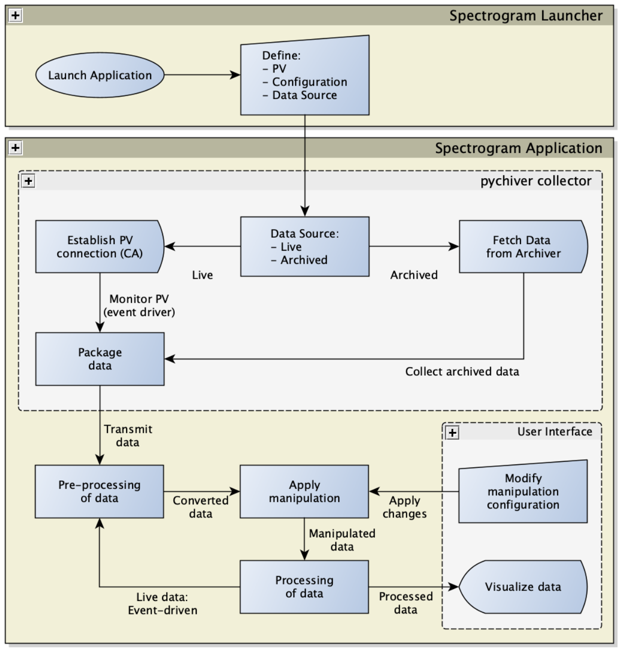

Figure 2: A high-level flowchart of the Spectrogram application utilizing a simplified model of the pychiver collector.

## FEATURES

Features include:

- Live manipulation of the data, including applying following to the data array:
  - Discrete Fourier Transformation
  - Inverse Discrete Fourier Transformation
  - Custom Discrete Fourier Transformation
  - Power Spectral Density
  - Data Filters
- Colormap setup
- A time grid (with distance between each grid object)
- Exporting and Importing data
- Time and Wave bins (less bins could yield higher performance in case processing power of the computer is exceeded)
- Region of Interest of the waveform

## SAMPLES

Over the span of the application being developed, it has been used in the ESS control room running on a MacBook Pro (connected via Wi-Fi to office network) to capture and visualize different events to better understand the physics behind them. Some examples are described beneath (following the quench of the SRF cavity, captured in Fig. 1).

### *Damping of the Adaptive Feedforward Table*

Whilst commissioning with a proton beam through the first buncher (an accelerating structure focusing the bunches in the direction they travel) of the medium-energy beam transport part of the linac, an adaptive feedforward was applied to ensure the correct cavity field and phase are applied to the beam within the cavity. When the beam was lost, the spectrogram application captured interesting wave-like ripples whilst monitoring the cavity field's phase, see Fig. 3.

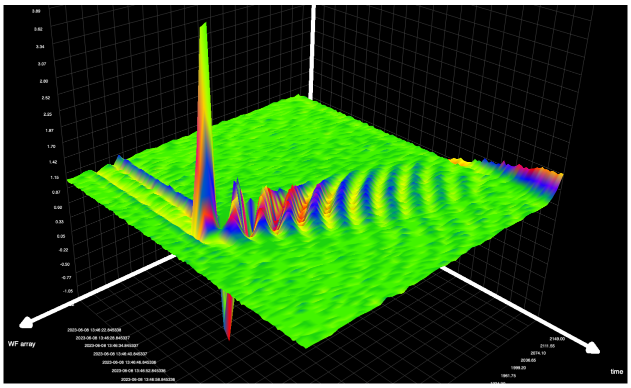

Figure 3: The buncher cavity field phase with adaptive feed forward before, during and after a beamloss event.

The cavity field phase is controlled by the low-level RF adaptive feed forward controller. Initial expectation would be a single peak remaining in the same position on each pulse but decreasing with the amount of pulses to reach the desired flat nominal cavity field. Since the adaptive feed forward is correcting on the error of the controller, the pattern showing that the peaks of the phase waveform array propagate along the time-axis within the pulse essentially move the error along the time-axis of each pulse. This could either be a natural effect of the adaptive feed forward or an indication that some parameters in the adaptive feed forward are not optimal. Further research is needed to fully understand this phenomena.

### *Increasing Proton Beam Current by Opening an Iris*

During proton beam commissioning up to Drift Tube Linac section's tank 4, the end destination of the beam is the Faraday Cup of this section. One does not start sending full beam current to the Faraday Cup, even if permits allow, as the safe practice is to start with a smaller current - which can be reduced using an iris located upstream along the linac. Gradually opening the iris then results in the gradual increase of the beam current, captured in Fig. 4.

As can be seen in Fig. 4, opening the iris shows that the longitudinal beam profile changes with the iris opening. It seems to indicate that it captures more high-energy protons (relative to average beam energy) as the amount of protons arriving at the beginning of each pulse relative to the rest of the pulse seems to increase with the iris opening.

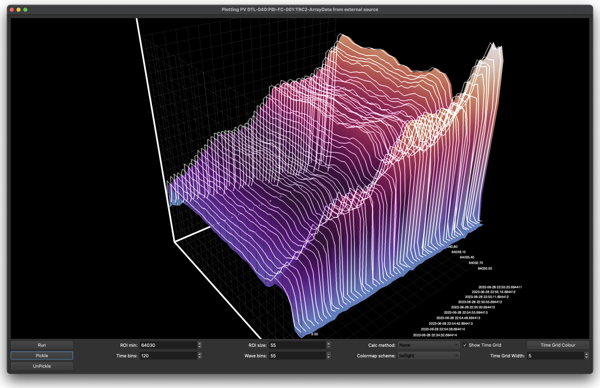

Figure 4: Current deposited on the Faraday Cup in Drift Tube Linac section 4 whilst the iris in the Low Energy Beam Transport section is opening.

### *Pulse Length Increment*

Whilst reconditioning the Radio Frequency Quadrupole (RFQ) it was observed that, at high power, power interlocks were triggered when increasing the pulse length. Therefore, the power was reduced to avoid triggering the interlocks during the investigation. Launching the Spectrogram revealed that each time the pulse length was increased, the power for some increment-regions (e.g. increasing from 2400 µs -> 2800 µs this region of 400 µs) initially had higher power levels than the rest of the pulse sees a higher power for the first pulse, captured in Fig. 5.

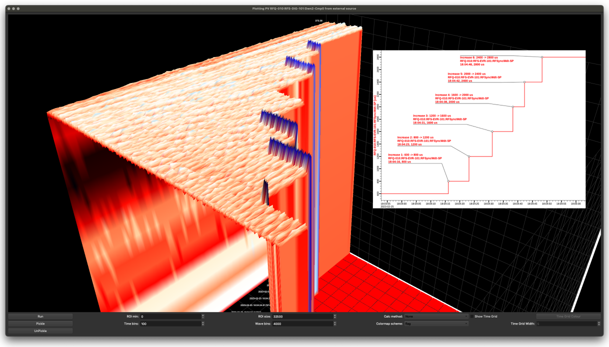

Figure 5: RFQ pulse length increase resulting in initially increased power for some incremented regions.

As can be seen in Fig. 5, the 3D waveform shows the power increase for each pulse length increase's increment region, which caused some power interlocks. It also showed that this did not occur for each pulse length increase, providing valuable input into understanding why the power interlocks occurred for pulse length increments.

## PERFORMANCE TESTS

The performance tests are set as benchmarking against the total number of events at 14Hz to check the amount of pulses captured and the visualization frame rate for different bin sizes and region of interests. The test system is an Apple MacBook Pro (2021, Apple M1 Max, 32GB).

A benchmark function was implemented that measures the amounts of waveforms received and compares this against the system's event receiver's cycle count, i.e. expected amount of waveforms transmitted from the system. It also measures the frames per second visualized. The results of 4 separate benchmarking tests, with two different array sizes and two different amount of bins, have been normalized and is shown in Fig. 6.

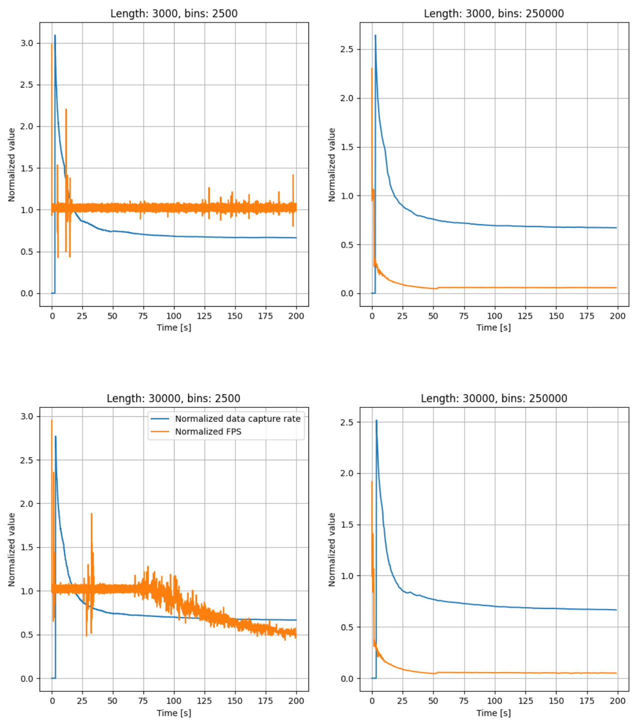

Figure 6: Benchmarking the Spectrogram application utilizing pychiver to establish how different data acquisitions rates and amount of bins impact the data capture and frames per second for the visualization.

As can be seen in Fig. 6, the results shows the following:

- Increased waveform size does not impact data capture, converging towards 67% with time for all test cases.
- For the large array small bin size, the frames per second decreases after ca 1.5min, suggesting that there are some issues with how the application handles buffering.
- The frames per second decreases rapidly with the amount of bins.

## CONCLUSION

Whilst the application is in a state ready to be used, further development is still needed. Functionalities needed for the 3D application includes a more stable implementation of the axis labels to ensure no overlap occurs, adding a axis calibrations (static or based on another waveform for the horizontal axis), the buffer function (see bullet point 2 in *Performance Tests*) and a function which separates the waveform in its current state into a new window (snapshot). The latter offers the advantage of allowing user to observe the current state in real time whilst applying manipulation to and investigating an event in a separate window.

Further development also includes adding more functionalities to the so-called 1D viewer with a direct switch between the different viewing methods, and to investigate whether this application can be implemented into a web page.

## ACKNOWLEDGEMENTS

The authors want to thank the Linac team members at ESS suggesting that it would be interesting to see a SRF quench using 3D visualization, which prompted a rapid development of the 3D functionality of the Spectrogram application in order to have it ready in time for the real event. The authors also want to thank the RF team and the Beam Physics team for being available for discussions on the physics of various events, including the ones in this paper.

## REFERENCES

[1] F. Léonard, "Phase spectrogram and frequency spectrogram as new diagnostic tools", *Mech. Syst. Signal Process.*, vol. 21, no. 1, pp. 125-137, 2007. doi:10.1016/j.ymssp.2005.08.011

[2] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA, USA: CreateSpace, 2009.

[3] *pyqtgraph*, O. Moore *et al.*, "PyQtGraph - High Performance Visualization for All Platforms", in *Proc. 22nd Python Sci. Conf.*, 2023, pp. 106-113. doi:10.25080/gerudo-f2bc6f59-00e

[4] PyQt5 Reference Guide, https://www.riverbankcomputing.com/software/pyqt/

[5] L. R. Dalesio *et al.*, "The experimental physics and industrial control system architecture: past, present, and future", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 352, pp. 179-184, 1994. doi:10.1016/0168-9002(94)91493-1

[6] PyEpics: Python Epics Channel Access, https://pyepics.github.io/pyepics/

[7] https://gitlab.esss.lu.se/accop/pytools/pychiver

[8] M. Woo *et al.*, *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*, Addison-Wesley Longman Publishing Co., Inc. 1999.

[9] C. R. Harris *et al.*, "Array programming with NumPy", *Nature*, vol. 585, pp. 357-362, 2020. doi:10.1038/s41586-020-2649-2