ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOSH4002

A CLOUD BASED FRAMEWORK FOR ADVANCED ACCELERATOR CONTROLS*

J. P. Edelen†, M. Keilman, P. Moeller, R. Nagler, RadiaSoft LLC, Boulder CO, USA

Abstract

Modern particle accelerator facilities generate large amounts of data and face increasing demands on their operational performance. As the demand on accelerator operations increases so does the need for automated tuning algorithms and control to maximize uptime with reduced operator intervention. Existing tools are insufficient to meet the broad demands on controls, visualization, and analysis. We are developing a cloud based toolbox featuring a generic virtual accelerator control room for the development of automated tuning algorithms and the analysis of large complex datasets. This framework utilizes tracking codes combined with with algorithms for machine drift, low-level control systems, and other complications to create realistic models of accelerators. These models are directly interfaced with advanced control toolboxes allowing for rapid prototyping of control algorithms. Additionally, our interface provides users with access to a wide range of Python-based data analytics libraries for the study and visualization of machine data. In this paper, we provide an overview of our interface and demonstrate its utility on a toy accelerator running on EPICS.

INTRODUCTION

As the demands on accelerator facilities increase so does the need for automated tuning algorithms and control to maximize uptime with reduced operator intervention. "More uptime means more physics" is a growing proverb for the industry aiming to optimize machine cycles and increase efficiency. One obstacle to these efforts is that existing tools are insufficient to meet the broad demands on controls, data visualization, and data analysis.

To meet these needs, we are building a web-based toolbox that features a generic virtual accelerator control room for the development of automated tuning algorithms and the analysis of large complex datasets. This package can be integrated with any accelerator control system to assist with complex data analysis tasks and development of control algorithms. This includes loading archive data and direct streaming of settings and readings from the control system to the toolbox. Our prototype interface is currently available from any web browser. Machine specific interfaces can be installed behind accelerator firewalls and accessed from anywhere on the controls network. Our toolbox bridges the gap between EPICS-based [1] control systems and in-house control systems by creating a common platform for analysis and simulation. By using a flexible software framework, we will have the ability to include machine learning libraries in the future. At this time, the computational power available to particle accelerator operations is increasing. The customer demands on the particle accelerator industry are expanding. Providing meaningful tools to the operators, scientists, and engineers will help particle accelerator facilities manage efficiency and ensure supply meets demand.

* This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under Award Number DE-SC0019682.
† jedelen@radiasot.net

A CLOUD-BASED DATA ANALYSIS TOOLBOX

Improved data analytics capabilities are essential for modern accelerator control rooms. Some facilities now incorporate jupyter notebooks into their daily operations with direct connections to machine parameters and data archiving tools. While this represents a significant step forward in accelerator operations, these modules can at times lack the intuitive feel of point-and-click analysis tools. As part of our framework we are developing a suite of analysis and visualization tools that allow users to upload, manipulate, and analyze data directly in an intuitive browser interface. Here we detail three primary features of our prototype toolbox and describe our plans for future work in this area.

Clustering

Clustering tools have been commonly used for many years, however their presence in accelerator analysis and controls has only recently seen wider adoption. scikit-learn [2] provides a simple gateway to a variety of clustering tools, however, users are required to build post-processing and cluster selection tools on their own. Our interface connects with four popular clustering algorithms and provides users with point-and-click operations on the dataset for cluster selection and further evaluation.

Figure 1 shows a sample dataset in two dimensions with results from running the clustering algorithm k-means. Users can select the clustering algorithms and modify the hyper parameters as needed. Users can click on different clusters and open them in a new plot which allows for further analysis to be performed. For example additional clustering, frequency analysis, or curve fitting.

Frequency Analysis

We have also implemented visualizations for numpy's frequency analysis tools [3]. Here users can easily perform a 1-D frequency analysis on imported datasets and identify the spectrum of a given input signal. Figure 2 shows an example of how the user interacts with the frequency analysis tools and visualizations.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOSH4002

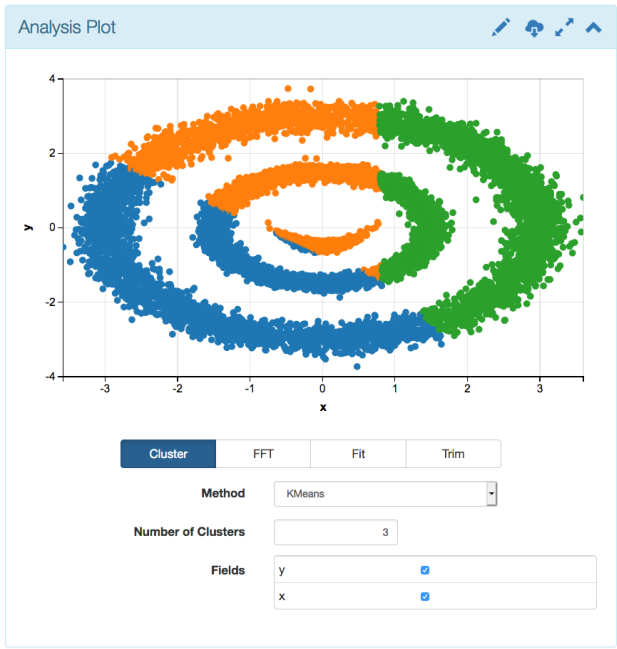

Figure 1: Screenshot of an example dataset clustered with k-means and colored by cluster number.

Our tool plots both the frequency spectrum and the original signal. Frequencies in the signal are identified and displayed on the top of the FFT plot for easy extraction. Users have the option to save the raw output from this analysis or images for presentations and reports.

Curve Fitting

Our interface also provides a compliment of curve fitting tools. Users can choose to fit either the whole dataset of a subset of the data through the use of the "trim" function. The trim function allows one to use point-and-click to extract a subset of data for further analysis. The equation should be entered using pythonic syntax for exponentials. The curve fitting parser utilizes SymPy [4] to build the fit function which allows for a wide range of functions and special functions to be implemented. Currently we support trigonometric functions. Independent and dependent variables are specified and a range of functions and combinations of functions can be used. Figure 3 shows a screenshot of what a user might see when trying to fit a series of sine functions to a dataset.

Here you can see equations displayed and confidence intervals for the fit parameters. These confidence intervals can be turned on or off depending on the desired visualization.

Future Plans

During the next stage of this project we will develop tools that allow users to analyze more than just 1-D features of datasets, for example 2-D histograms, plotting, and frequency analysis tools. We will also further our integration of these tools with accelerator control systems. We plan to connect these tools with the control system for the Collider Accelerator Division at Brookhaven National Lab. Additionally we will allow users to connect to remote EPICS servers and read parameter values both from data loggers or live from the control system.

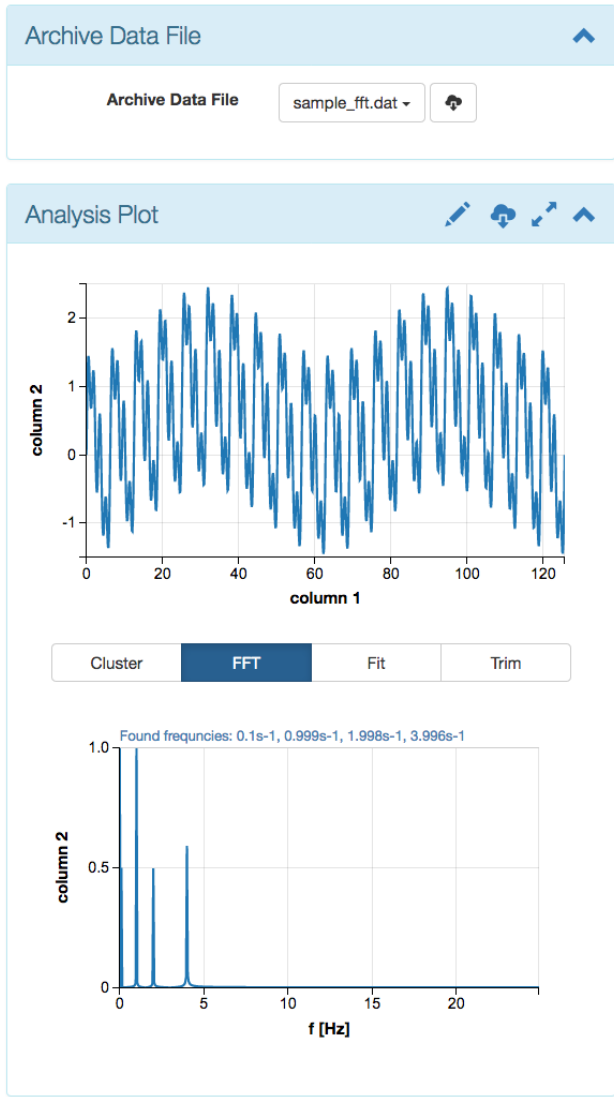

Figure 2: Screenshot of a frequency analysis of a test input signal described by $y = 0.5\sin(0.1x)+\sin(x)+0.5\sin(2x)+0.6\sin(4x)+0.5$

ACCELERATOR CONTROLS

In conjunction with the data analytics tools, we are also prototyping a high level controls toolbox that can connect both with accelerator control systems and with accelerator simulation tools. At present we are simulating a toy beamline implemented in elegant that is connected to a local EPICS database to mimic the environment of a real accelerator. Here we describe our interface and showcase a simple beam steering task and diagnostics currently available on our server.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOSH4002

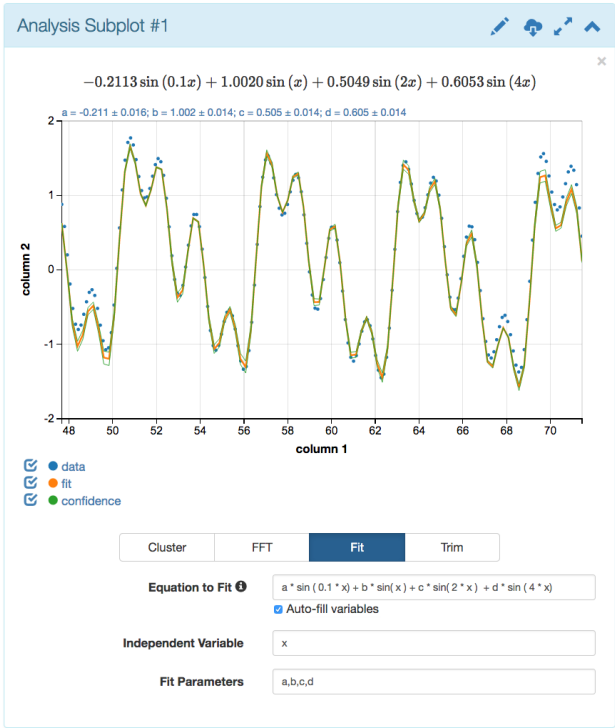

Figure 3: Screenshot showing a curve fitting example on a subset of the data shown in Fig. 2.

Local EPICS Server

Our local EPICS server is currently running EPICS version 7.0.2 with CA Protocol version 4.13. We have a database of parameters similar to what one would find in a real accelerator for a beam-line. In the background we have an elegant simulation running continuously by reading and writing to the EPICS database. The tuning algorithm and the Sirepo interface interact with the simulation through EPICS commands. In principle we could remove the beam-line model in elegant and replace it with a real machine and the functionality would be identical. Figure 4 shows a block diagram of the data flow for our test system.

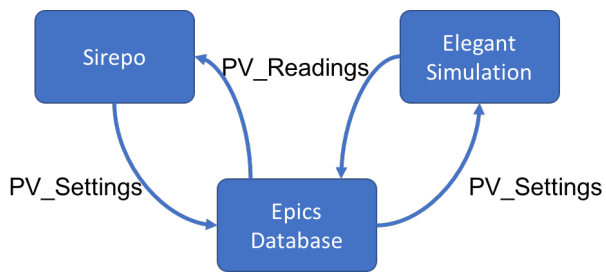

Figure 4: Block diagram of the signal chain for our local EPICS server and toy beam-line.

Toy Beam Steering Problem

To test our interface we implemented a simple beam steering problem using the Sirepo framework. Figure 5 shows the Sirepo landing page for the controls toolbox after we have initiated the connection with EPICS. At the moment the initial beam parameters are user inputs to allow for demonstration of the toolbox. The dropdown menu on the upper right hand side of the figure allows for different steering methods to be used. Currently we have a linear orbit correction or Nelder-Mead optimization. The Nelder-Mead optimization currently uses a hard coded cost function which seeks to minimize the position of the last two BPMs to ensure that the beam trajectory is as close to the center of the beam-pipe as possible.

The user sees an x-y plot for the beam position as well as the digital readout of the beam positions and the corrector settings. Quadrupoles are also accessible for more advanced tuning or for emittance scan automation, which will be added in the future. While the optimization runs, plots are updated in real time, and a history of the beam information is maintained, enabling users to interrupt at any point if the machine enters a bad state. Users can also view the beam position along the beam-line and a history of the position, shown in Fig. 6. Here the most recent set of beam positions are shown in purple, with greyscale representing time in the past.

Finally, users can view the corrector settings as a function of time, Fig. 7.

Future Plans

As part of our future plans these diagnostics will be generalized to allow for easy construction of custom interfaces for particular beam-lines. Users will be able to specify more advanced tuning algorithms and cost functions based on their individual needs and they will be able to share these interfaces by the simply copying and pasting a link into a different web browser. When operating outside accelerator firewalls, users will only have access to simulation data or archive data from a particular EPICS server.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under Award Number DE-SC0019682.

REFERENCES

[1] Experimental Physics and Industrial Control System, https://epics-controls.org/

[2] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825 – 2830, 2011.

[3] T. Oliphant, *NumPy: A guide to NumPy*, USA: Trelgol Publishing, http://www.numpy.org/

[4] A. Meurer *et al.*, "SymPy: symbolic computing in Python", *PeerJ. Computer Science*, vol. 3, p. e103. https://doi.org/10.7717/peerj-cs.103

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOSH4002

Figure 5: Screenshot of the accelerator controls landing page during a Nelder-Mead optimization.

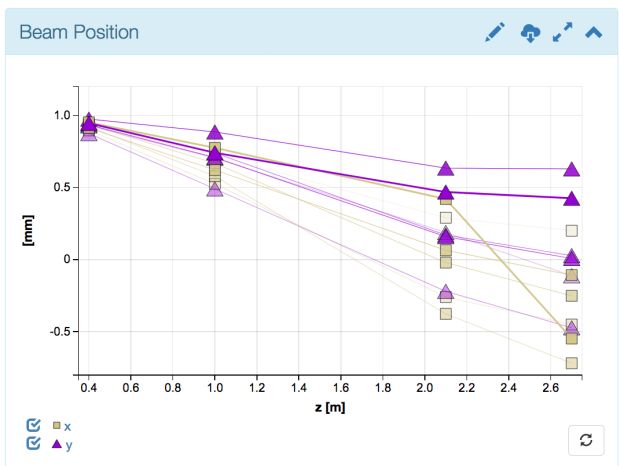

Figure 6: Screenshot of the BPM readings diagnostic showing the beam position along the beam-line.

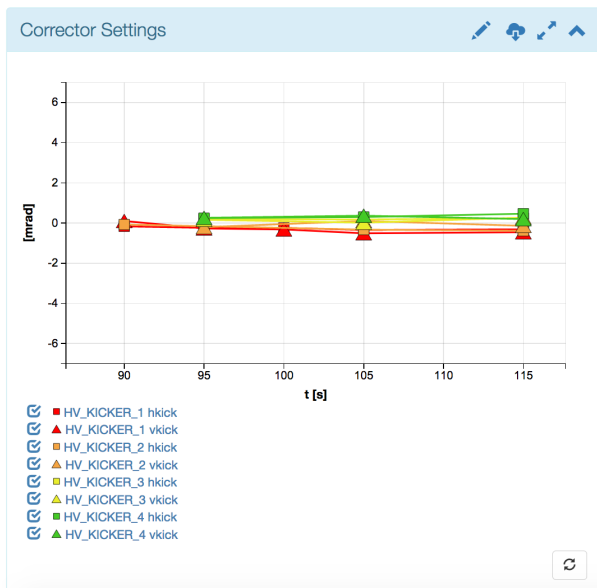

Figure 7: Screenshot of the corrector settings diagnostic showing the corrector settings as a function of time during an optimization.