High-level Physics Controls Applications Development for FRIB

Tong Zhang
Controls Physicist
Accelerator Physics Department
Facility for Rare Isotope Beams, Michigan State University
E-mail: zhangt@frib.msu.edu

Presented at 17th International Conference on Accelerator and Large Experimental Physics Control System
5-11 October 2019, New York, NY, USA

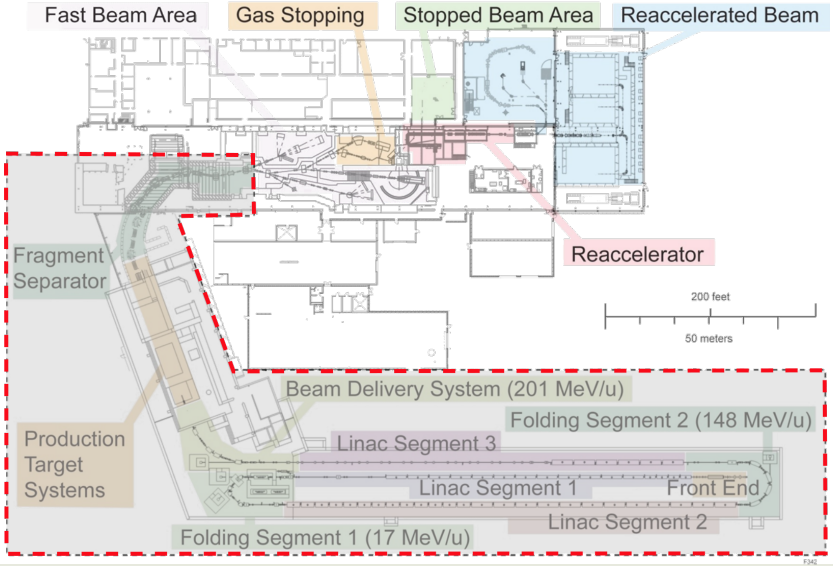
MICHIGAN STATE UNIVERSITY

U.S. DEPARTMENT OF ENERGY | Office of Science

This material is based upon work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661, the State of Michigan and Michigan State University. Michigan State University designs and establishes FRIB as a DOE Office of Science National User Facility in support of the mission of the Office of Nuclear Physics.

Outline

Outline


Fast Beam Area
Gas Stopping
Stopped Beam Area
Reaccelerated Beam
Fragment Separator
Reaccelerator
200 feet
50 meters
Beam Delivery System (201 MeV/u)
Folding Segment 2 (148 MeV/u)
Production Target Systems
Linac Segment 3
Linac Segment 1
Front End
Linac Segment 2
Folding Segment 1 (17 MeV/u)

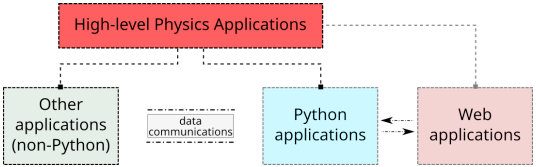
Accelerator System

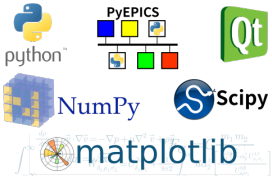
- particle source, beam transport, experimental end stations, ...
- devices: optics, diagnostics, ...
- distributed controls units (Channel Access): EPICS input & output controllers (IOCs)

High-level Physics Applications

- Final goal: operating accelerator facility
- Purpose: have robust and functional beam tuning algorithms
- Solution: software environment for high-level physics controls

High-level Physics Applications = Physics Algorithms + Controls Software





Fundamental Requirements

- Quick prototyping: dynamic programming language, powerful&friendly development environment
- Features: plenty of third-party packages
- End-users: good UI/UX with proven physics algorithms
- Agile development: develop → build → test → deploy

PHANTASY

Physics High-level Applications aNd Toolkit for Accelerator SYstem

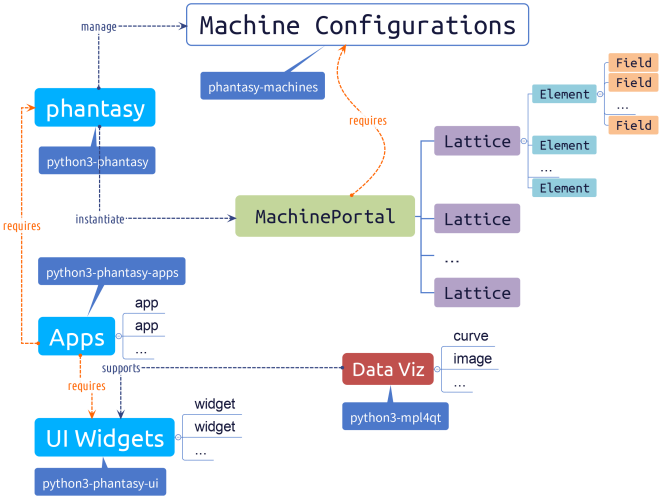
Features Highlight

- Device configuration management: maintainability, portability
- Device abstraction: object-oriented
- Online modeling: physics model-depends machine representation
- Python interactive scripting environment for high-level controls: development and control
- Virtual accelerator based on EPICS and physics model: test physics algorithms
- Web service integration (channelfinder, scanserver, UNICORN ...): extendability

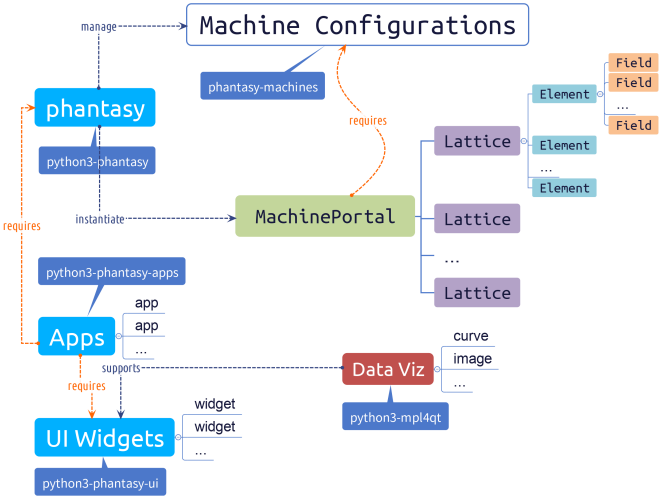
Deployment

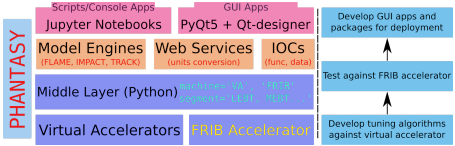
- Target OS: Debian 8, 9, 10
- Meta package: phantasy (~20 packages)
- Physics model engines: FLAME, IMPACT, TRACK

Architecture of PHANTASY Project



Architecture of PHANTASY Project





Code Facts

- Python 2 and 3, Debian OS
- phantasy (core) ~19k LOC
- UI widgets: ~11k LOC (incl. dataviz, excl. ui)
- Apps: ~26k LOC (excl. ui)

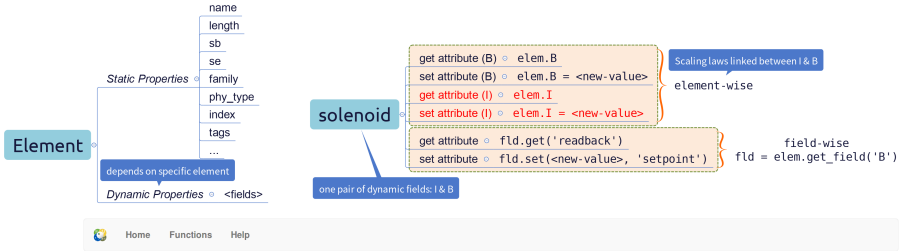
Dev Guideline

- Device ⟶ Python object
- List of devices ⟶ Lattice
- Lattice manipulation
- Scripting/GUI apps (facility-agnostic)
- Package ⟶ deployment

Device Abstraction

- Standardize device description data file
- Handle EPICS Channel Access (CA) in OO level
- Benefits: WORA, facility-agnostic script/apps

- Manage physics and engineering units by dedicated web app
- REST APIs for units interpretation
- Benefits: availability and extendability



Read/write policy from/to device could be defined with REST APIs.

▹ WORA: Write Once, Run Anywhere.

Interactive Python Scripting Environment

```
In [1]: from phantasy import MachinePortal
        from phantasy import disable_warnings
        disable_warnings()

In [2]: mp = MachinePortal("FRIB", "LEBT")

In [3]: mp
Out[3]: [FRIB] MachinePortal | Valid segment: *LEBT, MEBT, LEBT2FS1A, LEBT2FS1B, MEBT2FS1A, MEBT2FS1B, LINAC

In [4]: lat = mp.work_lattice_conf

In [5]: layout = lat.layout

In [6]: fig1, ax1 = layout.draw(span=(1.05, 1.1), fig_opt={'figsize': (20, 8), 'dpi': 120}, ax_opt={'aspect': 6})
```


```
In [7]: lat
```

Out[7]:

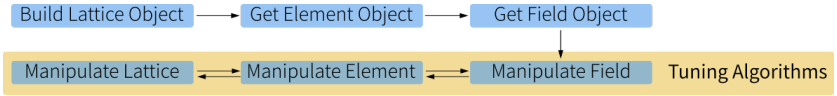
	Name	Family	Position	Length
0	FE_SCS1:SOLR_D0704	SOL	70.2069949964	0.3998
1	FE_SCS1:DCV_D0709	VCOR	70.9469949964	0.0
2	FE_SCS1:DCH_D0709	HCOR	70.9469949964	0.0
3	FE_SCS1:DH_D0717	BEND	71.1819949964	1.0
4	FE_SCS1:DCV_D0723	VCOR	72.3441949964	0.0
5	FE_SCS1:DCH_D0723	HCOR	72.3441949964	0.0
6	FE_SCS1:QHE_D0726	EQUAD	72.5292949964	0.2052
7	FE_SCS1:QVE_D0730	EQUAD	72.864294996039999	0.2052
8	FE_SCS1:QHE_D0733	EQUAD	73.1992949964	0.2052
9	FE_SCS1:FC_D0739	FC	73.7692949964	0.0

```
In [8]: sol = mp.get_elements(type='SOL')[0]

In [9]: sol
Out[9]: FE_SCS1:SOLR_D0704 [SOL] @ sb=70.206995
```

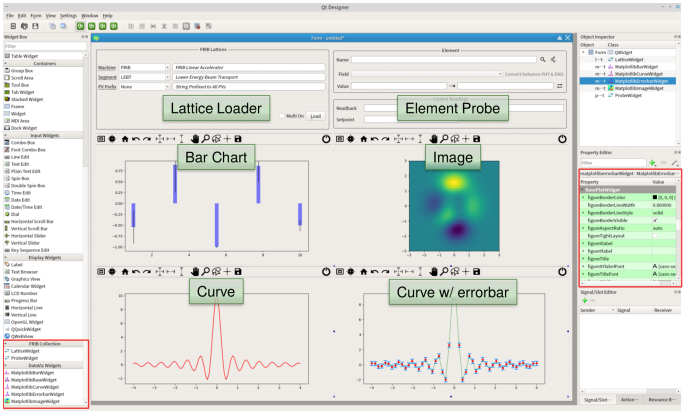
70	FE_LEBT:FC_D0977	FC	97.66932979	0.0
71	FE_LEBT:DCV_D0979	VCOR	97.68935	0.0
72	FE_LEBT:DCH_D0979	HCOR	97.68935	0.0
73	FE_LEBT:SOLR_D0982	SOL	98.0101	0.3998
74	FE_LEBT:PM_D0986	PM	98.629999	0.0
75	FE_LEBT:CAV_D0987	CAV	98.735	0.0
76	FE_LEBT:BCM_D0989	BCM	98.881492	0.0
77	FE_LEBT:DCV_D0992	VCOR	99.19935	0.0
78	FE_LEBT:DCH_D0992	HCOR	99.19935	0.0
79	FE_LEBT:SOLR_D0995	SOL	99.3201	0.3998
80	FE_LEBT:FC_D0998	FC	99.7955237	0.0
81	FE_LEBT:PM_D0998	PM	99.8138409	0.0

```
In [8]: sol = mp.get_elements(type='SOL')[0]

In [9]: sol
Out[9]: FE_SCS1:SOLR_D0704 [SOL] @ sb=70.206995

In [10]: sol.I
Out[10]: 0.1

In [11]: sol.B
Out[11]: -0.0002898000000000083

In [12]: sol.I / sol.B
Out[12]: -346.02076124567475

In [17]: sol.B = -0.5

In [18]: sol.I
Out[18]: 173.01038862283735

In [ ]:
```

Continue with more complex work...



Outline




```
tong@zhangt-mint ~$ makeBasePyQtApp
usage: makeBasePyQtApp [-h] [--app APP] [--template TEMPLATE]
                       [-l [LIST_TEMPLATES]]

Initialize PyQt5 app project based on app template (--template), pass --app
with the name of application to generate files in the current working
directory, or with the full path to generate to any other directory.

optional arguments:
  -h, --help            show this help message and exit
  --app APP             path of app to create
  --template TEMPLATE   name of app template
  -l [LIST_TEMPLATES]   list all available app templates
```

- Standardized UI style (package: phantasy-ui)
- Standradized app development workflow
- Publish-ready quality figure
- High productivity
- Facility-agnostic apps
- Streamline deployment (package: phantasy-apps)

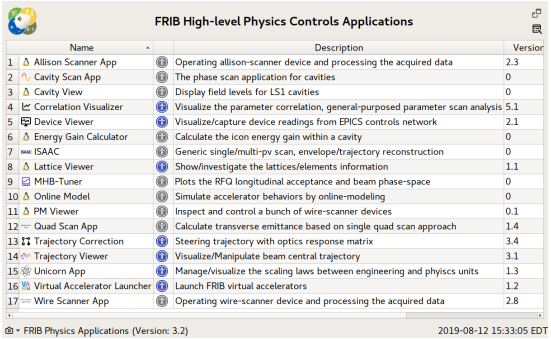
- FRIB Collection: lattice, element, etc.
- DataViz Widgets: curve, curve w/ errorbar, image, etc.
- Commands: run_designer, frib_designer, makeBasePyQtApp

Package: mpl4qt

- ✓ Toolbar system
- ✓ Configuration system
- ✓ Qt-designer widgets

▷ mpl4qt: Data visualization widgets implmeneted with matplotlib for Qt5.

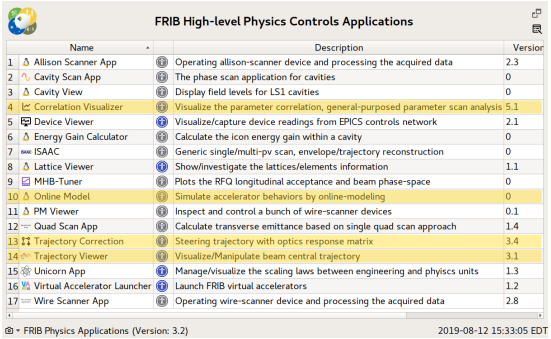
App Launcher: Manage All Apps in One Place



Features

- Global entrypoint: right-clicking context menu → ‘Phyapps Apps’
- Configuration file (.ini) controlled for easy maintaining
- Run app by double-clicking, w/ or w/o console, keep logs or not

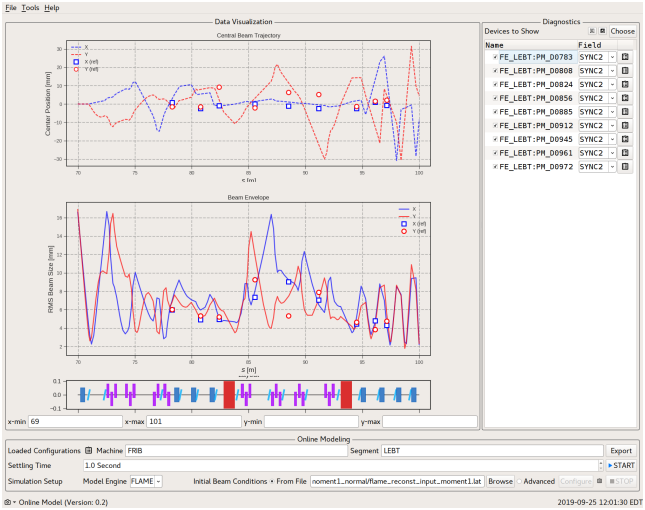
App Launcher: Manage All Apps in One Place

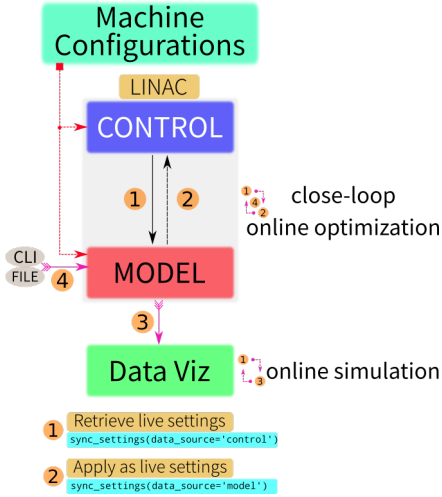


Features

- Global entrypoint: right-clicking context menu → ‘Phyapps Apps’
- Configuration file (.ini) controlled for easy maintaining
- Run app by double-clicking, w/ or w/o console, keep logs or not

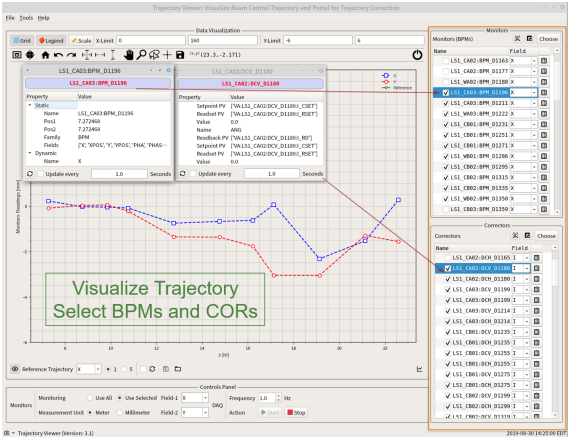
Accelerator Online Simulation: Predict Live Physics Performance





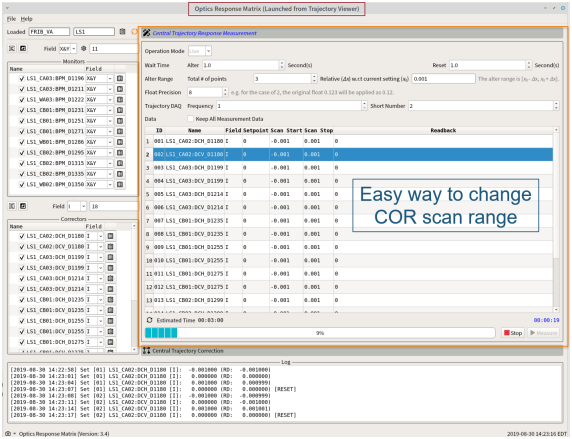
▷ FLAME: Fast Linear Accelerator Model Engine, envelope simulation code (C++/Python) developed at FRIB.

Online Central Trajectory Correction: Based on response matrix



Trajectory Viewer App

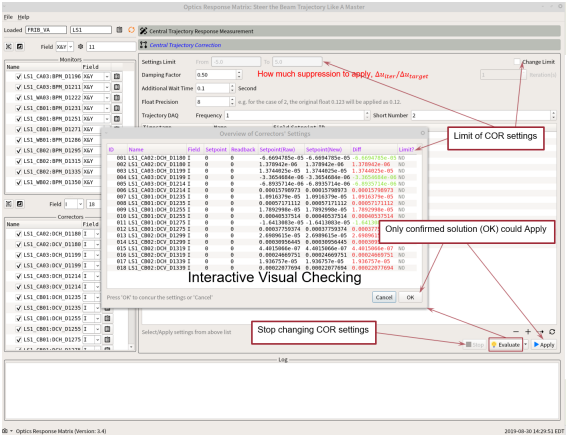
- Load lattice, Tools → Load Lattice
- Select BPMs and Correctors
- Visualize trajectory

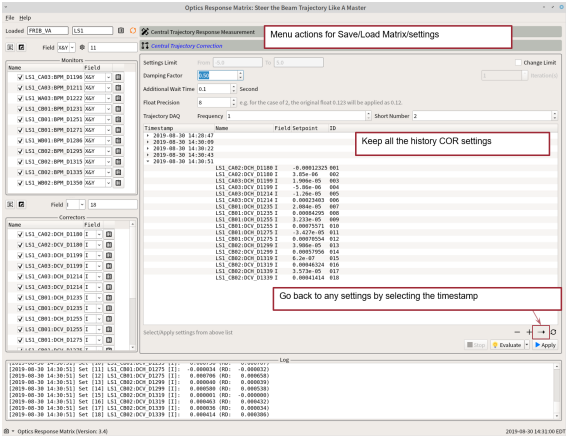


Optics Response Matrix App

- Set up correctors alter ranges
- Measure optics response matrix, save/load
- Correct trajectory with optics response matrix

Online Central Trajectory Correction: Based on response matrix

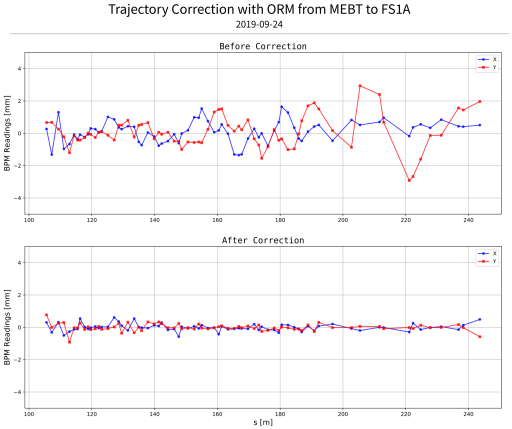


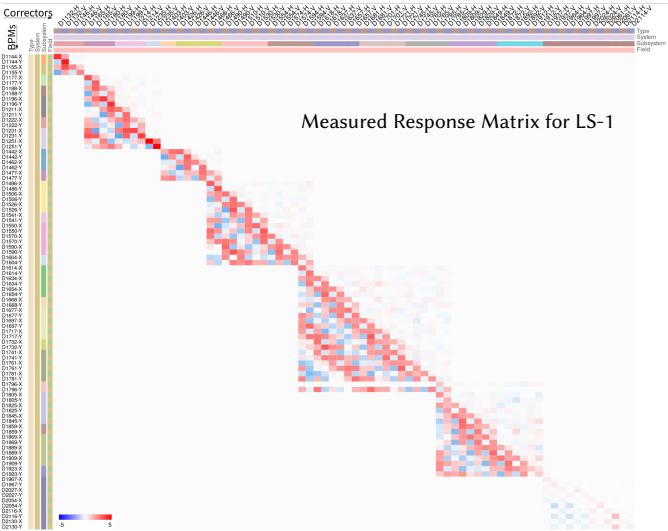


- Gradually suppress trajectory
- Secure tuning range of correctors
- Visual confirmation for each solution
- Stop button

- Keep all the history settings of correctors
- Go back and forth at any history settings
- Save/load settings, matrix

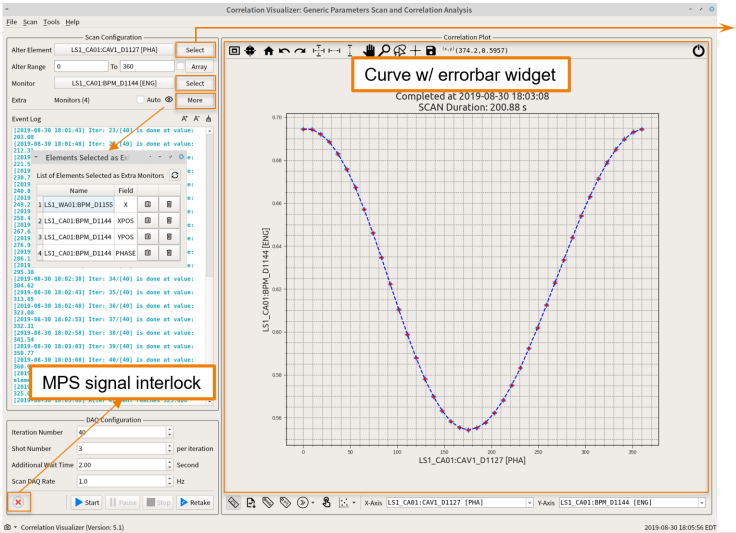
Online Central Trajectory Correction: Based on response matrix



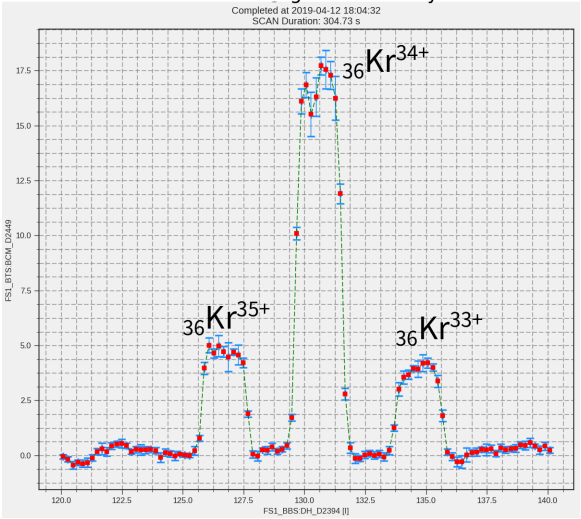


General Parameter Correlation Analysis and Visualization

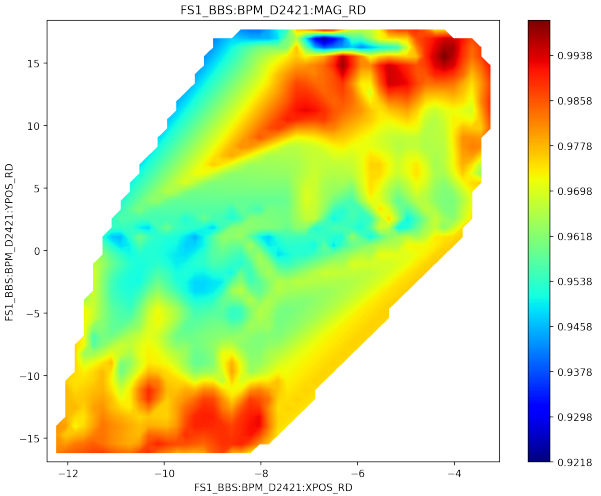
Scan parameters by mouse clicking



Multi-charge states study



Extend to Higher Dimensions



Outline

- Developed Python-based software framework (PHANTASY) for high-level physics controls at FRIB
- PyQt5 based modularized generic GUI apps are developed for FRIB
- Efficient machine tuning is achieved with the developed apps for FRIB LINAC commmissioning
- Apps based on PHANTASY also work with other EPICS-based facilities, no phyiscs model dependency
- Development of physics model progresses as the commissioning advances

---

Future Development Plan

DATA	Unify and standardize data interface of all physics apps, streamline the data post-processing
GUI	Develop GUI toolkit for high-level physics apps development by any developer
MODEL	Based on the data flow from PHANTASY framework, refine physics model by machine-learning
COLLABORATION	https://github.com/phantasy-project/

Acknowledgements

Dylan Maxwell, Martin Konrad, Daron Chabot, Eric Berryman, Steven Beher, Masanori Ikegami, Bruno Martins, Diego Omitto, Scott Cogan, Steven Lidia, Michael Davidsaver, Guobao Shen, Kei Fukushima, Tomofumi Maruta, Takashi Yoshimoto, Jonathan Wong, Alexander Plastun, Qiang Zhao, Yue Hao, Steve Lund, Peter Ostroumov

Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661, the State of Michigan and Michigan State University.

Thank you for your attention!