DEVELOPMENT STATUS OF FRIB ON-LINE MODEL BASED BEAM COMMISSIONING APPLICATION

Z. He*, K. Fukushima, D. Maxwell, M. Davidsaver, Y. Zhang, G. Shen, Q. Zhao
FRIB, Michigan State University, USA

Abstract

The new software FLAME has been developed to serve as physics model used for on-line beam commissioning applications. FLAME is specially designed to cover FRIB modeling challenges to balance between speed and precision. Several on-line beam commissioning applications have been prototyped based on FLAME and tested on the physics application prototyping environment. In this paper, components of the physics application prototyping environment are firstly described. Then, the design strategy and result of the four major applications: baseline generator, cavity tuning, orbit correction, transverse matching, are discussed.

INTRODUCTION

Facility for Rare Isotope Beams (FRIB) is a new national user facility for nuclear science co-funded by DOE and MSU [1, 2]. In order to handle the special modeling challenges of FRIB driver linac [3–5] while keeping fast speed for on-line purpose, an envelope-tracking based code, the Fast Linear Accelerator Modeling Engine (FLAME) [6] , is developed to provide useful tools for the challenges of FRIB linac modeling and tuning.

The physics application prototyping environment is built for physicists to easily build and test various modeling and tuning applications [7]. The detailed structure of this physics application prototyping environment is discussed in first section. After that, several major physics applications to support FRIB beam commissioning, such as baseline generator, cavity tuning, orbit correction, transverse matching, are developed and discussed in the following section.

PHYSICS APPLICATION PROTOTYPING ENVIRONMENT

The infrastructure of the physics application prototyping environment is described in Fig. 1. A distributed infrastructure is utilized with python to be the middle layer scripting language. A virtual accelerator driven by IMPACT [8] or FLAME is built mimicking the behavior of a real accelerator. The virtual accelerator is wrapped by the same EPICS [9] layer as real accelerator, so that the physics application developed for a virtual accelerator would be directly usable for a real accelerator. The FLAME model service is built onto the python middle layer via its native python interface. Two major kinds of optimizers, Design Analysis Kit for Optimization and Terascale Applications (Dakota) [10] optimizer and Scipy [11] optimizer, are implemented as the optimizer

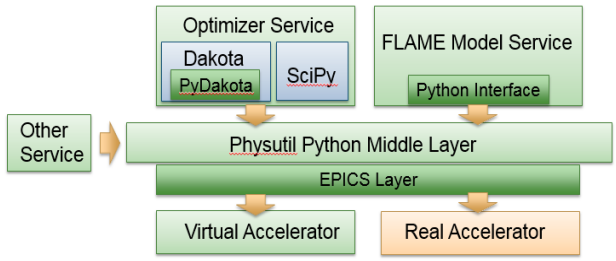


Figure 1: Infrastructure of the physics application prototyping environment.

service module. Dakota contains algorithms for optimization with gradient-based local and non-gradient-based local/global methods, written in C++ with a pyDakota [12] python interface. It also has useful features such as hybrid minimization method and native support for MPI parallelization. Scipy also has optimization capability inside package Scipy.optimize, they are easy to use and offers a different set of optimization methods, and can be a good alternative choice. The Dakota optimizer is chosen as the main optimizer in our following discussion. The distributed infrastructure would also allow easy access to any other service needed such as scan service and unit conversion.

MAJOR PHYSICS APPLICATIONS PROTOTYPES

In this section, four major physics applications developed with the physics application prototyping environment are discussed, namely baseline generator, cavity tuning, orbit correction and transverse matching.

Baseline Generator

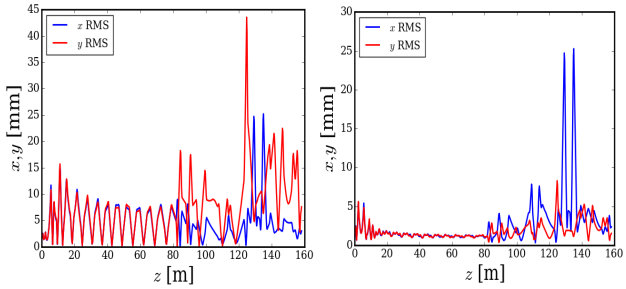
Before beam tuning of FRIB driver linac, we always need a baseline as our tuning goal. Even though a baseline lattice already exists, this design may not be enough to guide FRIB beam commissioning task due to unexpected element fault, change of lattice design requirement, or numerous other situations where the baseline lattice has to be redesigned within a short period of time. Baseline generator is specially designed to handle this problem.

Baseline generation can be reduced to an optimization problem. With FLAME model service and optimizer service in place, all we need is to specify the tuning nob, tuning range, optimizing method and tuning goal. One specific use case is discussed below.

Transverse recovery: in this case, we assume the third superconducting solenoid of LS1 gets failed, causing transverse beam envelopes to blow up as Fig. 2(a). Then, the

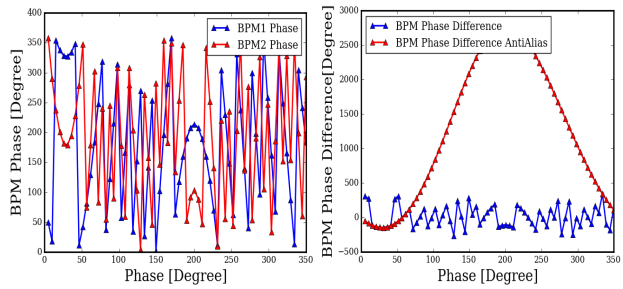
* hez@frib.msu.edu

ISBN 978-3-95450-169-4



(a) (b)

Figure 2: Transverse RMS recovery from solenoid fault. (a) transverse RMS size blows up after the third superconducting solenoid of LS1 is tripped; (b) transverse RMS recovery;



(a) (b)

Figure 3: Demonstration of FLAME-based cavity tuning application. (a) raw phase reading of two downstream warm BPM of a target RF cavity; (b) calculation of the BPM phase difference and 2-pi anti-aliasing.

solenoids around the broken solenoid are chosen as tuning nobs. The conjugate gradient (CG) optimization method is used. Minimization of the average transverse rms profile is set as our optimization goal. After optimization from the code, new settings of solenoids are obtained to recover the lattice without the failed solenoid (Fig. 2(b)).

Cavity Tuning

The purpose of cavity tuning application is to set the phase and amplitude of RF cavities with design values using downstream BPMs [13].

After chosen a certain RF cavity and two downstream BPMs (for demonstration, we choose the first superconducting RF cavity and first and second warm BPMs in Linac Segment 1), we first do 2-pi phase scan of the RF cavity and then gather two sets of downstream BPM phase readings. The cavity amplitude is treated as unknown, and should be calculated using this application. The raw data is provided by the virtual accelerator, which is shown in Fig. 3(a) Then, we calculate the BPM phase difference and reconstruct the sine curve-like data using 2-pi anti-aliasing technique (Fig. 3(b)). Next, we try to reproduce the sine-like curve using FLAME by fitting three parameters: initial kinetic energy, cavity phase and cavity amplitude. After fitting the three parameters, we can calculate the settings of phase, RF amplitude and beam kinetic energy to be 349.599^o, 64.065% of nominal working voltage and 0.499977 MeV/u, while the design is 349.741^o, 64.0%, 0.50 MeV/u. The algorithm is making very precise prediction.

One challenge for cavity tuning using BPM phase reading comes from the $2-\pi$ ambiguous phenomenon. The phase reading from two warm BPMs is always within $2-\pi$, while the real accumulated phase difference between two warm BPMs is actually $2N\pi+\Delta\phi$, where N is unknown and cannot be directly got from BPM phase reading. However, assume that the kinetic energy is consistent with model, N can be calculated exactly form on-line model. For example, according to FLAME calculation, for the first cavity and first plus second warm BPMs, N can be as large as 53. That means if beam speed drifts 2% away from design, we'll probably guess a wrong number of N, resulting a gap between real energy and energy estimation, which can potentially mess up the cavity tuning of the whole linac. As aresult, cross-validation should be added to prevent this situation to occur.

Orbit Correction

The main goal of orbit correction application is to correct the beam orbit at each BPM using steerers. In this study, we first introduce unknown random misalignment to all the RF cavities, solenoids, quadrupoles in LS1 in both horizontal and vertical directions into our virtual accelerator. The goal is to minimize the relative readings of all BPMs. Two methods: traditional model-based orbit response matrix (ORM) with singular value decomposition (SVD) method [14] and FLAME-based optimization method, have been tested and proved to be effective.

The algorithm of FLAME-based ORM is described as below. After getting the orbit response matrix with FLAME, we can calculate kick angle of correctors (Eq. (1)):

$$\vec{\theta} = \sum_{\alpha=1}^{N} \frac{C_\alpha}{\omega_\alpha} \bar{\tau}^{(\alpha)} \tag{1}$$

Where $C_\alpha = \vec{d} \cdot \bar{\delta}^{(\alpha)}$, $\vec{d}$ is BPM correction vector calculated from difference between BPM readings and target,and $\bar{\delta}^{(\alpha)}$, ω_α, $\bar{\tau}^{(\alpha)}$comes from singular value decomposition of orbit response matrix A (Eq. (2)):

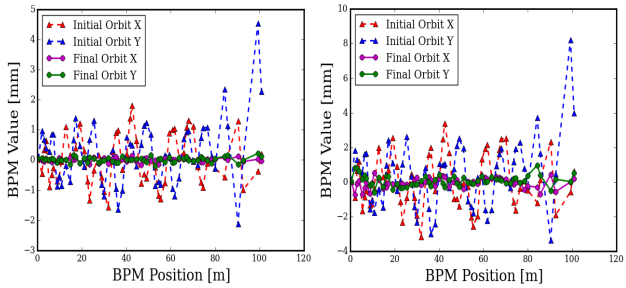
$$A = DWT^t \tag{2}$$

D is the M-by-N matrix whose column vectors $\bar{\delta}^{(\alpha)}(\alpha = 1,\ldots,N)$ form an orthonormal set. W is N-by-N diagonal matrix with non-negative elements. T^t is the transpose of the N-by-N matrix T, whose column vectors $\bar{\tau}^{(\alpha)}(\alpha = 1,\ldots,N)$ also form an orthonormal set.

For FLAME-based optimization method, we choose all correctors in LS1 as tuning nobs and set tuning range according to the maximum and minimum correction angle. The cost function we are going to minimize is Eq. (3):

$$M = |\delta\bar{d} + \alpha(\bar{d}_c - \bar{d}_t)|^2 \tag{3}$$

ISBN 978-3-95450-169-4



(a) (b)

Figure 4: An example of orbit correction, (a): ORM-based orbit correction, (b): optimizer-based orbit correction.

where $\delta\bar{d}$ means change of BPM readings calculated by FLAME due to change of corrector kick angle. Similarly, α is damping factor between 0 1, $\bar{d}_c$ is current BPM displacement reading, $\bar{d}_t$ is target BPM reading. Because around a hundred corrector parameters are supposed to be optimized simultaneously, gradient-based local method (in our case CG) is usually used.

An example of final correction result is plotted in Fig. 4. Figure 4(a) shows the result with ORM and Fig. 4(b) shows the result with optimizer-based orbit correction. Both method can significantly suppress the orbit distortion caused by random lattice element misalignment. And ORM is converging faster than optimizer-based orbit correction.

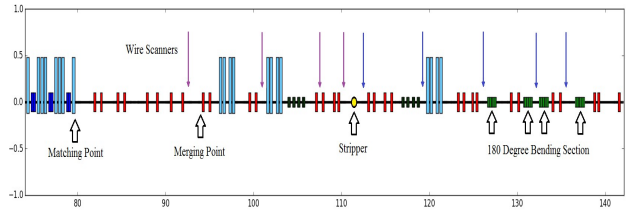
Transverse Matching

The main purpose of transverse matching application is to match transverse beam envelope to be the same as baseline design using input from wire scanners [15, 16]. The following example demonstrate matching of beam from solenoid lattice segment into quadrupole lattice segment. We assume round beam and negligible horizontal-vertical beam coupling.

The matching strategy can be divided into three steps: Step 1: some mismatch is intentionally introduced by arbitrary adjusting some solenoids upstream in virtual accelerator. Step 2: get wire scanner profile monitor (PM) readings and calculate the CS-parameter at matching point (shown in Fig. 5). We choose $[\beta_x, \alpha_x, \epsilon_x, \beta_y, \alpha_y, \epsilon_y]$ at matching point as tuning nob and use FLAME to calculate the horizontal and vertical RMS beam size at downstream wire scanner position. The calculated rms beam size is then compared with the wire scanner reading, and the difference is minimized using optimizer. Step 3: use quadrupoles between matching point and merging point to adjust the CS-parameter at merging point back to design. The matching result is shown in Fig. 6. We can see that the algorithm described above successfully brought beam RMS wire scanner readings back to design, indicating successful beam matching.

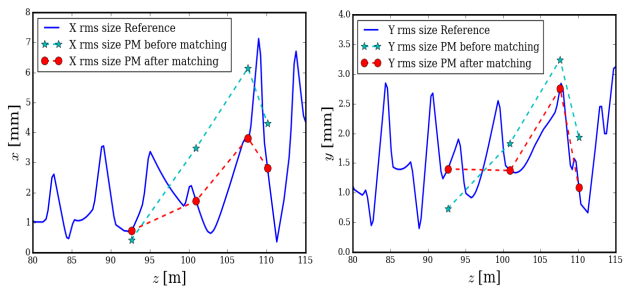
CONCLUSION

A new software FLAME has been developed for the purpose of FRIB on-line model service and physics application. In order to support physics application development, the physics application prototyping environment, including VA layer, EPICS layer, python middle layer, model service and optimizer service, has been built up. Major physics applications, such as baseline generator, cavity tuning, orbit correction and transverse matching, have been prototyped.



(a)

Figure 5: Schematic plot of the lattice of matching section between solenoid-focusing lattice and quadrupole-focusing lattice.Lattice legend: blue: solenoid; cyan: RF cavity; red: quadrupole; green: dipole; yellow: stripper; black: drift. Magenta arrow: wire scanner before stripper; blue arrow: wire scanner after stripper.



(a) (b)

Figure 6: FLAME based transverse matching result. Blue line: baseline RMS profile; cyan star: wire scanner reading before matching; red star: wire scanner reading after matching. (a) horizontal matching result; (b) vertical matching result.

ACKNOWLEDGMENT

The author would like to thank F. Marti, S. Lund, T. Maruta, T. Yoshimoto and M. Ikegami for helpful discussions and suggestions. The work is supported by the U.S. National Science Foundation under Grant No. PHY-11-02511, and the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661.

REFERENCES

[1] J. Wei, D. Arenius, E. Bernard, N. Bultman, F. Casagrande, S. Chouhan, C. Compton, K. Davidson, A. Facco, V. Ganni, *et al.*, "The FRIB Project–Accelerator Challenges and Progress," in *Proc. HIAT'12*, Chicago, USA. paper MOB01, pp. 8–19.

[2] J. Wei, D. Arenius, N. Bultman, F. Casagrande, C. Compton, K. Davidson, J. DeKamp, B. Drewyor, K. Elliott, A. Facco,

et al., "Progress towards the facility for rare isotope beams," in *Proc. NA-PAC'13*, *FRYBA3*, 2013.

[3] H.-D. Betz, "Charge states and charge-changing cross sections of fast heavy ions penetrating through gaseous and solid media," *Reviews of Modern Physics*, vol. 44, no. 3, p. 465, 1972.

[4] M. Cavenago, "Determining the radiofrequency dipole and quadrupole effects in quarter wave resonators," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 311, no. 1, pp. 19–29, 1992.

[5] A. Facco and V. Zvyagintsev, "Beam steering in superconducting quarter-wave resonators: An analytical approach," *Physical Review Special Topics-Accelerators and Beams*, vol. 14, no. 7, p. 070101, 2011.

[6] M. Davidsaver, J. Bengtsson, Z. He, K. Fukushima, G. Shen, and M. Ikegami, "Flame documentation," `http://frib-high-level-controls.github.io/FLAME/`, 2016.

[7] G. Shen, E. Berryman, Z. He, M. Ikegami, D. Liu, D. Maxwell, and V. Vuppala, "Physics Application Infrastructure Design for FRIB Driver Linac," in *Proc. ICALEPCS2015*, pp. 962–965, paper WEPGF113, `doi:10.18429/JACoW-ICALEPCS2015-WEPGF113`, 2015.

[8] J. Qiang, R. D. Ryne, S. Habib, and V. Decyk, "An object-oriented parallel particle-in-cell code for beam dynamics simulation in linear accelerators," in *Proceedings of the 1999 ACM/IEEE conference on Supercomputing* ACM, 1999, p. 55.

[9] L. R. Dalesio, M. Kraimer, and A. Kozubal, "Epics architecture," in *Proc. ICALEPCS'91*, pp.278–281, 1991.

[10] B. M. Adams, W. Bohnhoff, K. Dalbey, J. Eddy, M. Eldred, D. Gay, K. Haskell, P. D. Hough, and L. Swiler, "Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.0 user's manual," *Sandia National Laboratories, Tech. Rep. SAND2010-2183*, 2009.

[11] E. Jones, T. Oliphant, P. Peterson, *et al.*, "Open source scientific tools for python," 2001.

[12] N. R. E. Laboratory, "Dakota python interface pydakota," `https://github.com/NREL/DAKOTA_plugin/tree/master/pyDAKOTA`, 2014.

[13] Y. Zhang, P. Chu, and Z. He, "Phase and Amplitude Tuning Algorithms for the FRIB Superconducting Cavities," in *Proc. IPAC'15*, paper MOPWI025, pp. 1207-1210, `doi:10.18429/JACoW-IPAC2015-MOPWI025`, 2015.

[14] Y. Chung, G. Decker, and K. Evans, "Closed Orbit Correction Using Singular Value Decomposition of the Response Matrix," in *Proc. PAC'93*, 1993, pp. 2263–2265.

[15] H. Sako, A. Ueno, T. Ohkawa, Y. Kondo, T. Morishita, M. Ikegami, H. Akikawa, *et al.*, "Transverse Beam Matching and Orbit Corrections at J-PARC LINAC," in *Proc. LINAC'08*, Victoria, BC, Canada, paper MOP078, 2008.

[16] Y. Zhang, P. Chu, Z. He, S. Lund, and D. Maxwell, "Transverse Matching of Horizontal-Vertical Coupled Beams for the FRIB Linac," in *Proc. IPAC'15*, paper MOPWI025, pp. 1207-1210, `doi:10.18429/JACoW-IPAC2015-MOPWI025`, 2015.

ISBN 978-3-95450-169-4