

ONLINE ACCELERATOR MODELING WITH TWO CONTROLS SYSTEMS AT FACET-II

Z. Buschmann*, M. Gibbs, N. Majernik, R. Loney, G. Yocky
SLAC National Accelerator Laboratory, Menlo Park, CA, USA

Abstract

FACET-II is a unique experimental facility housed in 1 km of the original Stanford Linear Accelerator tunnel. Multiple generations of hardware are still in use, as are two generations of software controls. A majority of subsystems (RF, magnets, timing, etc.) have their controls split across both ecosystems. Three software layers, including a newly-developed online modeling infrastructure, bridge this gap to form a unified high-level abstraction of the accelerator used for data analysis and as a foundation to develop control-room physics applications. We discuss the implementation of this infrastructure and some downstream programs.

ACCELERATOR CONTROLS

Electron beam system controls at FACET-II [1] are split across two systems. A legacy mainframe controls large sections of the main linac, while EPICS [2] is used for the photoinjector, early linac and two bunch compressors. The experiment beamline at the accelerator front-end has a mixture of devices controlled by both systems. Figure 1 shows the structure and approximate jurisdictions of these systems. Relevant legacy software is defined below.

SLC Control Program (SCP): Custom-built accelerator controls system [3], hosted by an openVMS mainframe called “MCC” (Main Control Computer), which communicates with an array of microcluster computers (or “micros”) via a custom FM cable network called SLCNet to control accelerator hardware.

Accelerator Independent Data Access (AIDA-PVA): Java process on MCC that handles read and write requests to the SCP and MCC database over the SLAC controls network using the PVAccess protocol.

Channel Access Repeater: Java process on MCC that broadcasts a limited set of values for read-only access at up to 1 Hz using the Channel Access protocol.

LIVE MODEL ENGINE

Consistent accelerator operations depend on having access to a descriptive model of extant machine conditions. At SLAC, a number of programs have been written to read magnet and RF settings from the machine to update a model for performing calculations. With an API for communicating with accelerator controls, it is possible to write a script that will read current settings to update a model. In the SCP these computations were run directly on the mainframe by a model job.

* zack@slac.stanford.edu

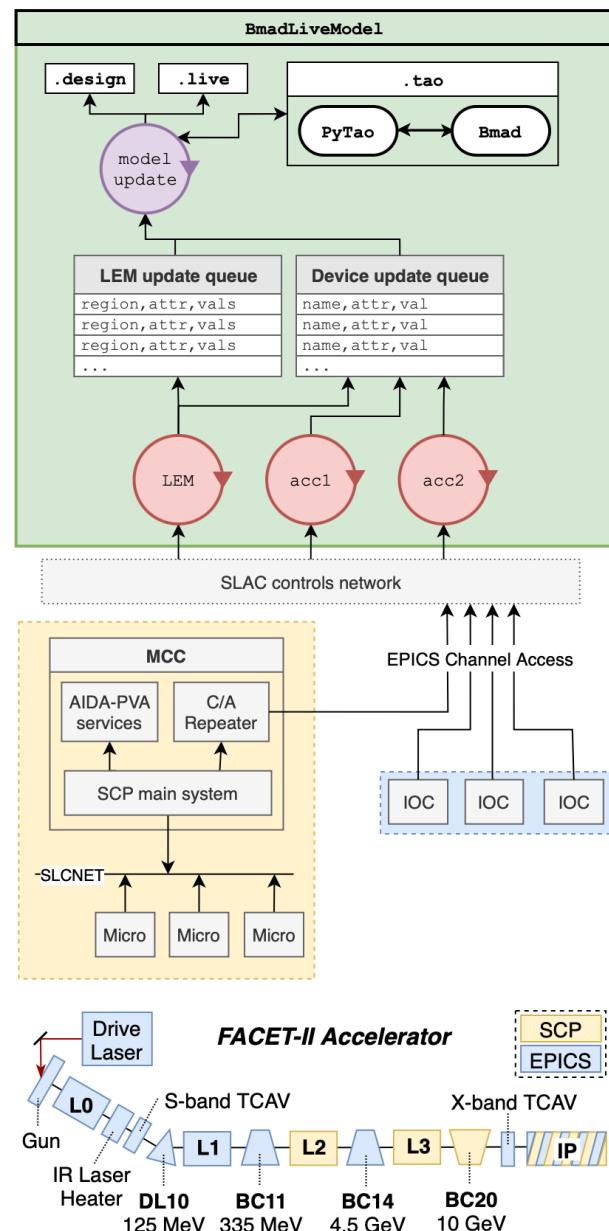


Figure 1: (Upper) Architecture of the BmadLiveModel class. (Lower) Schematic of the FACET-II accelerator and controls systems, with approximate jurisdictions labeled.

We have built a live modeling “engine” to solve this problem more generally than uses a thread-based architecture to collect accelerator controls system information and update a locally instantiated modeling code. This system is portable, outperforms simpler scripts and is designed to streamline

analysis and application development. It forms the foundation of a high-level beam-based physics software ecosystem used for daily operations of FACET-II.

Our modeling codes of choice are Bmad, Tao and PyTao; chosen for their rich documentation, active user community, and quantity of expert resources at SLAC [4, 5].

Architecture

The live model program is implemented with the BmadLiveModel Python class [6] (see Fig. 1). This class has two principal responsibilities: (1) device monitoring, and (2) updating the local instance of Bmad/PyTao. For performance, these tasks are separated into threads. There are three *watcher* threads reporting configuration changes from the controls system. The other thread is the *model update* task, which submits those configuration changes to PyTao and updates user-facing data structures.

Mediating communication between these threads are two *update queues*. The first is for single-device updates, to which all watchers send requests for element parameter changes. The second is a special *LEM queue* that handles global data.

The machine settings needed for simple linear optics and charged-particle simulations are: (1) the actual beam kinetic energy profile $p_z(s)$, which depends on what complement of RF cavities¹ are being used to accelerate the beam and (2) magnet strength settings.

The most critical dependency of the software is the map from elements in Bmad (i.e. those in a MAD deck or other design source), to names that can be used to identify corresponding devices in the field. For this purpose, the element *alias* field in Bmad is populated with an associated EPICS record name in the production lattice files [7].

Device Monitors

The first two watcher threads, named *acc1* and *acc2*, mainly update accelerator magnet settings. *acc1* monitors quadrupoles, while *acc2* handles solenoid, dipole and sextupole magnets, as well as miscellaneous devices such as movable collimators. These tasks enforce error tolerances and handle unit conversions before submitting update requests to the queue.

Linac Energy Manager (LEM)

The third watcher is the LEM thread, which monitors RF amplitude and phase distribution to appropriately set cavity parameters for modeling. This task estimates the live momentum of the beam, as well as magnet settings so that it remains matched to the accelerator lattice.

The beam momentum is usually a monotonically increasing function of s , determined by the current RF configuration of the linac. At a given position s , the beam momentum $p_z(s)$

is (approximately):

$$p_z(s) \approx \sum_i^{N_c(s)} A_i \cos(\phi_i) \quad (1)$$

where $N_c(s)$ counts the number of RF cavities up to the position s_i . A_i and ϕ_i are the amplitude and phase of each cavity as reported by the controls system².

Due to resolution limits and measurement error, the estimated $p_z(s)$ can differ from the beam energy measured by a dipole spectrometer. Since the actual momentum $p_{z,0}$ at this location is known, errors upstream are corrected with a “fudge” factor.

$$p_{z,\text{actual}}(s) = K_{\text{LEM}} p_z(s), \quad K_{\text{LEM}} = \frac{p_{z,0}}{p_z} \quad (2)$$

Tao Update Manager

The final thread is the model-update process. At each iteration, this task will empty both update queues, and format and send batch update commands to Tao. It will also push updated values to python data structures. During normal operation this process typically executes $O(10 - 100)$ updates per iteration. Since the underlying simulation is still running in Fortran, iteration times are $O(100 \text{ ms})$.

API

Commonly accessed data are accessible as Numpy [8] arrays. Model data structures are categorized as either static values (element names, s-positions, etc.), design values (beam momentum, Twiss parameters, magnet settings), and extant values. The local PyTao instance is also publicly accessible, with its own API that binds directly to Tao as well as the ability to pass and run arbitrary Tao command line instructions.

PHYSICS APPLICATIONS

Live Model Service

The initial design use case for the BmadLiveModel is as the backend of a server that provides real-time model data over the controls system. Two separate PVAccess server processes using a BmadLiveModel instance publish (1) LEM data and Twiss parameters and (2) transport maps, respectively. Separation of these tasks improves performance and reliability.

LEM Matching Application

With the best-guess estimate of $p_z(s)$, the LEM watcher calculates the beam rigidity along the machine and suggested magnet settings using the design multipole strength coefficient k_n and effective magnet lengths l_{eff} from the model.

$$B_{\text{LEM}} = (B\rho)k_nl_{\text{eff}}$$

² The cavities, RF and high power distribution schemes at FACET-II have been well characterized. The controls system can measure (with $\approx 1\%$ accuracy) the effective energy gain applied to the beam, ignoring beam loading.

¹ The FACET-II linac is driven by 69 total SLAC model 5045 S-band klystrons, of which only ≈ 60 are needed to achieve a beam energy of 10 GeV.

In the control room, a GUI is used to display recommended B_{LEM} settings and trim magnets accordingly.

Final Focus/Interaction Point Optics

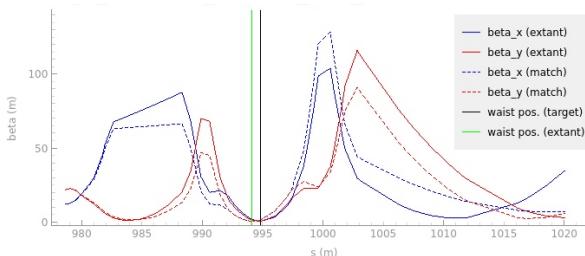


Figure 2: Optimization result of a request to shift the final focus waist upstream.

FACET-II has a tunable final focus, so that the highest quality beam parameters can be delivered to different locations in the IP region. BmadLiveModel runs optimizations via a GUI to find quadrupole field strengths that produce desired Twiss values at arbitrary s locations in the IP (see Fig. 2). Tao optimizations run in bound Fortran libraries, and thus usually take less than 1 s.

Beam-Based Longitudinal Feedbacks

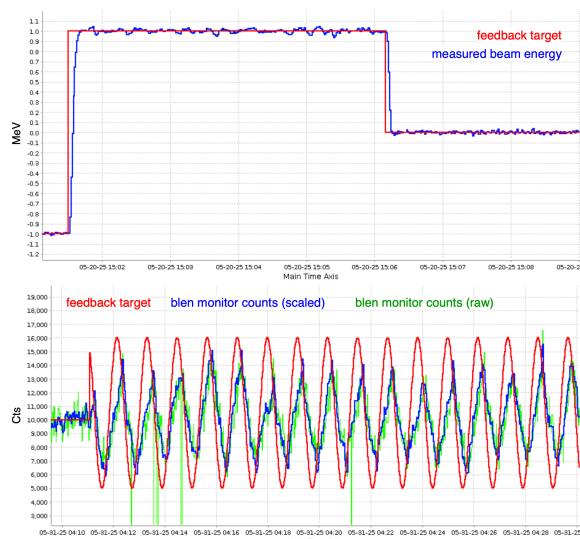


Figure 3: (Upper) Beam energy feedback responding to a setpoint change. (Lower) Driven sinusoidal oscillation of the beam current during an experiment.

Stable beam operations are achieved through beam-based feedback systems (see example in Fig. 3). The maximum repetition rate of FACET-II is 30 Hz, and carefully designed Python code can easily operate synchronously. However, measurements provided over the Channel Access repeater

are limited to 1 Hz, and AIDA-PVA requests similarly take $O(700 \text{ ms})$. This places a fundamental limit on system performance, since the maximum disturbance frequency a PID controller can reject is proportional to its sample rate. As a result, SCP-mediated loops have regulation bands (in percent) that are around 1 order-of-magnitude larger than those in the EPICS-only areas.

CONCLUSION

We have developed a live modeling infrastructure for FACET-II that is portable and agnostic to its bifurcated controls system. It is currently in use in multiple control-room physics tools. The design is intentionally portable, which we will put to the test by next porting this system to the LCLS facility. This work also ties into ongoing efforts to implement a near-real-time phase space reconstruction [9] system at SLAC.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge B. Ripman and our colleagues in the accelerator operations group for extensive feedback during application development and testing. C. Zimmer and E. Yang also contributed valuable program design consultation. This work was supported by the U.S. Department of Energy under DOE Contract No. DE-AC02-76SF00515.

REFERENCES

- [1] G. R. White, “A design for 10 gev, high peak-current, tightly focused electron beams at FACET-II”, in *Proc. IPAC’17*, Copenhagen, Denmark, May 2017, pp. 807–809. doi:10.18429/JACoW-IPAC2017-MOPIK115
- [2] L. Dalesio, M. Kraimer, and A. Kozubal, “EPICS Architecture”, <http://www.aps.anl.gov/epics/>
- [3] R. Melen, “Design and performance of the Stanford Linear Collider control system”, *IEEE Trans. Nucl. Sci.*, vol. 32, p. 230, 1985. doi:10.1109/TNS.1985.4336826
- [4] D. Sagan, “Bmad: A relativistic charged particle simulation library”, *Nucl. Instrum. Meth.*, vol. A558, no. 1, pp. 356–359, 2006. doi:10.1016/j.nima.2005.11.001
- [5] C. Mayes, D. Sagan, and K. Lauer, “PyTao”, 2018. <https://github.com/bmad-sim/pytao>
- [6] Z. Buschmann, “FACET-II live model infrastructure”, 2024. https://github.com/slaclab/F2_live_model
- [7] M. Woodley, G. White, and M. Ehrlichmann, “FACET-II lattice”, 2020. <https://github.com/slaclab/facet2-lattice>
- [8] C. R. Harris *et al.*, “Array programming with NumPy”, *Nature*, vol. 585, no. 7825, pp. 357–362, 2020. doi:10.1038/s41586-020-2649-2
- [9] R. Roussel *et al.*, “Phase space reconstruction from accelerator beam measurements using neural networks and differentiable simulations”, *Phys. Rev. Lett.*, vol. 130, no. 14, p. 145 001, 2023. doi:10.1103/PhysRevLett.130.145001