01: Introduction to Beam Simulations

WONG, Chun Yan Jonathan; HAO, Yue;
LUND, Steven; RICHARD, Christopher;

USPAS Accelerator Physics

June 2018

(Version 20180606)

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.

Numerical Modeling in Accelerator Science

- Numerical modeling with computers is an essential part of scientific research
  - Insight
  - Complement theory and experiment
    - Explore inaccessible parameters
    - Apply to systems too complicated for pure theoretical approach
  - Graphical display & visualization
    - Can measure non-interactively to improve insight

- Many types of computational studies employed in accelerator science and engineering
  - Beam dynamics
  - Electromagnetic and static modeling of lattice elements
  - Electron / ion sources
  - Diagnostic data analysis
  - Others: beam-matter / beam-plasma interactions etc.

Uses of Beam Simulations

- Lattice design & choices
- Guide experiments
- Probe tolerances of mechanical misalignments, field errors, …
- Machine tuning & performance optimization
- Diagnose failures and problems
- Check theory
- Discover new phenomena

Limitation of Modeling

- Can fail / misguide when needed physics not included
- Algorithms can fail when misused
- Some problems require large computational resources

Hierarchies of Beam Simulation Codes

- Envelope codes: beam represented by statistical moments
  - Functional form of beam distribution assumed to be unchanged
  - Used in rapid machine design / tuning
  - E.g. elegant, TRACE3D, MAD-X, TRANSPORT, …

- Particle codes: beam represented by ensemble of particles
  - Can be self-consistent
  - Can require large resources, but capable of high detail
  - E.g. elegant, Warp, IMPACT, PyORBIT, TRACK, …

- Some codes can do the work of both
  - E.g. elegant, PARMILA, …

Envelope Codes – Design & Tuning

- Fast
  - E.g. FLAME at FRIB front end: each run takes 20 ms!
    - » Routinely makes thousands of runs to probe errors and optimize
- Design and tuning
  - Orbit correction
  - Parameter optimization
  - Constraint fitting
- Usually employs linear optics
- Many effects not modeled or greatly simplified
  - Typically lacks full self-consistency

Particle Codes – Beam Dynamics Studies

- Slow (mostly)
- Track particles throughout beamline, single- or multi-pass
  - Can register where particles are lost
  - Dynamic aperture and long-term stability
  - Allows detailed comparison with diagnostic measurements
- Can employ more accurate beamline model
  - Import realistic representation of external fields
  - Can simulate non-EM elements (collimators, energy degraders, strippers)
- Can include beam EM radiation effects
- Can include collective phenomena
  - Space charge (beam self-fields)
  - Impedances (beam-induced EM fields in accelerator elements)
  - Electron clouds (electrons trapped by ion beam potential)
  - Beam-beam effects (colliders)

Choosing a Code

- Code must include needed physics
  - Run speed can be an issue
  - Obtaining / installing
  - Ease of use

- Large codes have many options, choices must be made correctly!
  - Lattice element models
  - Treatment of collective phenomena
  - Turning effects on and off
  - Numerical algorithms

- Code should be benchmarked
  - Code vs. existing codes on well-established cases
  - Code vs. analytic solution or limiting cases
  - Code vs. experiments

This Class Uses the Code elegant

- Developed and maintained by Argonne National Lab (chief architect: Michael Borland)
  - Download: https://www.aps.anl.gov/Accelerator-Operations-Physics/Software#12345
  - Manual: https://ops.aps.anl.gov/manuals/elegant_latest/elegant.html
  - Users forum: https://www3.aps.anl.gov/forums/elegant/

- Suits the needs of our class
  - Large range of model options covering what we discuss in the course
  - Work as both envelope and particle code
  - Freely distributed, multi-platform
  - Widely used, handy addition to your toolkit

- Use free RadiaSoft cloud implementation
  - Simplify setup: no installation needed
  - Use only browser interface

Conventional Installation & Execution

- Significant effort to install and ensure package compatibility.
  - Different for each platform
  - E.g. Elegant: https://www.aps.anl.gov/Accelerator-Operations-Physics/Software
- Long learning curve to run simulations and process results
  - Example commands to run elegant and process results:
    - » elegant ring.ele
    - » sddsplot -col=s,betax -col=s,betay par.twi
  - Example input files:

Elegant lattice file

```
! *** Define Elements ***

! Quads
Q1: quad,l=0.25,k1=1.0
Q2: quad,l=0.25,k1=-1.0

! Drifts
D0: drift,l=0.5
D1: drift,l=2.0

! Bend
B1: sbend, l=1.0,angle=0.314159265,e1=0,e2=0


! *** Build Beamline ***

BL: line=(D0,Q1,D0,Q2,D1,B1,D1)
RING: line=(20*BL)

```

Elegant command file

```
&run_setup
        lattice = example.lte,
        default_order = 2,
        use_beamline = RING,
        rootname = example,
        final = %s.fin,
        p_central = 107.6
&end

&run_control
&end

&bunched_beam
        n_particles_per_bunch = 10000,
        one_random_bunch=1,
        emit_x = 4.6e-8,
        emit_y = 4.6e-8,
        beta_x = 10, alpha_x = 1,
        beta_y = 10, alpha_y = 1,
        sigma_dp = 0.001,
        sigma_s = 650e-6,
        distribution_type[0] = 3*"gaussian",
        distribution_cutoff[0] = 3*3,
        symmetrize = 1,
        enforce_rms_values[0] = 1,1,1,
&end

&track
&end
```

Cloud Implementation of elegant Provided Freely by RadiaSoft

- RadiaSoft: http://radiasoft.net/
- Code installed on RadiaSoft servers
  - Access using HTML5 compatible browser interface
- 1) Sirepo
  - https://beta.sirepo.com/#/elegant
  - More user-friendly setup via GUI
  - Post-processing tools readily available
- 2) Python wrapper
  - Implemented using Jupyter notebook
  - Use Python to generate input files and execute runs
  - Post-processing with Python graphics tools

What does it mean to execute a code "in the cloud"

- Cloud computing is a buzzword, and will probably fade in time
  - used to be called "client-server"
  - then it was called "software as a service" or SaaS
  - for a short while, everyone talked about "grid computing"

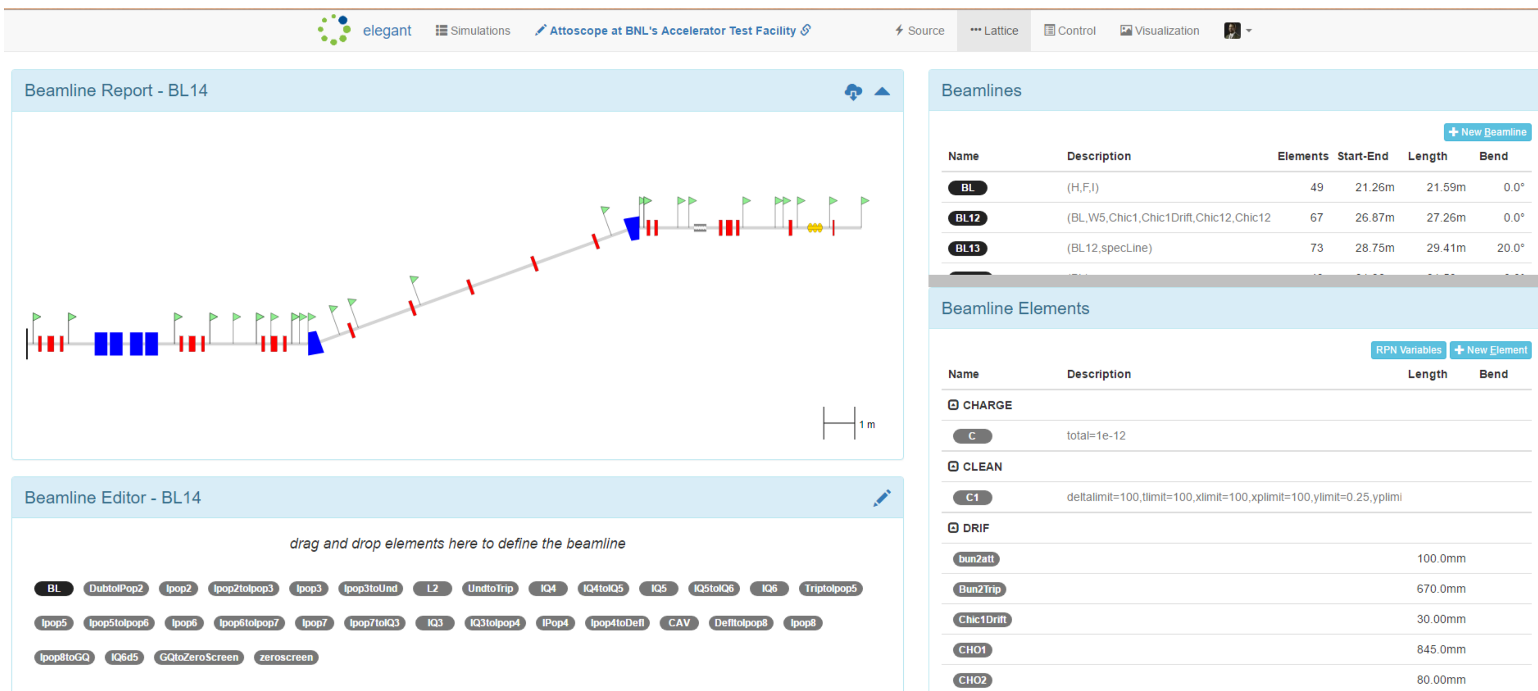
- The physics code is running on a remote "server"
  - probably running on Linux, possibly on a cluster or supercomputer
  - might be on "bare metal", such as your institution's cluster down the hall
  - might be running on a commercial cloud provider, like AWS

- The UI is your computer browser
  - whether you are banking, shopping, or designing a linac

- This wasn't practical 5+ years ago, so what changed?
  - the HTML5 standard was adopted by all modern browsers
    - » **the same GUI can now function well in any modern browser on any OS**
  - the JavaScript language (nothing like Java) emerged as a standard
    - » **many powerful JavaScript libraries and frameworks became available**
  - browsers have become powerful precompilers for executing code

The Sirepo cloud computing framework

- Open source, https://github.com/radiasoft/sirepo
- Freely available in open beta, https://sirepo.com
- Growing number of codes
  - X-ray optics: SRW, Shadow
  - Particle accelerators: elegant, Warp (special cases), more on the way
- Growing number of users
  - independent servers at BNL/NSLS-II, LBNL/ALS and PSI/ETH Zurich
  - about 100 users visit the open beta site



Two-level Structure: Compiled Code Linked to Flexible Interpreter

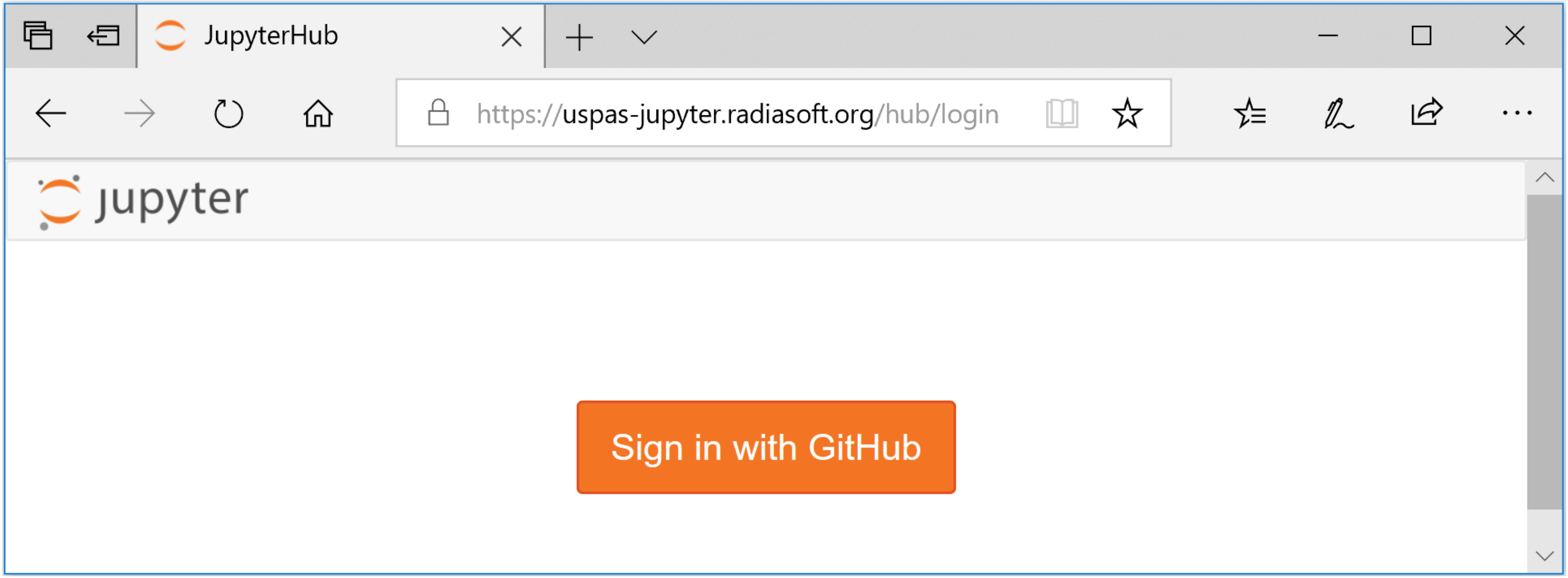
- Compiled code (e.g. FORTRAN, C, C++, etc.) at lower level for fast computation
  - Numerically intensive operations (e.g. particle movers, filed solvers)
- Scripting language (e.g. Python) at upper level for ease of organization
  - Beamline setup
  - Run configuration
  - Diagnostics
- Allows flexible use of code
- We will use a “light” 2-level model
  - Python scripts to setup elegant runs
  - Python scripts to process results

Why Python?

- Easy to read and learn
- Fully object-oriented design
- Open-source, with huge and supportive community
- Numerous packages to extend core Python
  - Numpy: manipulation of numerical arrays
  - Scipy: scientific computation
  - MatPlotLib: plotting
  - Pandas: data analysis
  - Many more
- Integration with compiled languages
  - C / C++ most natural
  - FORTRAN possible
- Tools to develop graphical user interfaces readily available
  - E.g. PyQt, wxPython, PyGUI, …

RadiaSoft JupyterHub

- Create Github account if you do not have one
- Sign in to JupyterHub on Radiasoft: https://uspas-jupyter.radiasoft.org/



Utilizing Git (First Time)

- Open terminal:

  - Terminal is bourne shell using standard UNIX commands for file manipulations

- First time: clone github repository

  >>> git clone https://github.com/YueHao/USPAS_AP_ComputerLab.git

```
jupyter$ pwd
/home/vagrant/jupyter
jupyter$ git clone https://github.com/YueHao/USPAS_AP_ComputerLab.git
```

  - Generates directory USPAS_AP_ComputerLab containing course examples / exercise

- Recommends updating files using git before each exercise (see next slide)

Utilizing Git for Updates

- Obtain updates from github repository:
  - Go to directory USPAS_AP_ComputerLab

```
jupyter$ pwd
/home/vagrant/jupyter
jupyter$ cd USPAS_AP_ComputerLab/
USPAS_AP_ComputerLab$ pwd
/home/vagrant/jupyter/USPAS_AP_ComputerLab
USPAS_AP_ComputerLab$
```

>>> git pull

  - If you have modified the files, git pull may see a conflict
  - Use these commands (may wipe out local changes, backup if necessary)

>>> git fetch --all

>>> git reset --hard origin/master

- Advise: copy files out of USPAS_AP_ComputerLab and work there
  - Example: create directory “LabExercise” and copy “IntroPython.ipynb” there

```
jupyter$ mkdir LabExercise
jupyter$ ls
LabExercise  Untitled  USPAS_AP_ComputerLab
jupyter$ cd USPAS_AP_ComputerLab
USPAS_AP_ComputerLab$ cp IntroPython.ipynb ../LabExercise/
USPAS_AP_ComputerLab$
```

Advice on Avoiding Merge Conflicts

- Copy files from USPAS_AP_ComputerLab to a new directory LabExercise
  - Example: create directory “LabExercise” and copy “IntroPython.ipynb” there

```
jupyter$ mkdir LabExercise
jupyter$ ls
LabExercise  Untitled  USPAS_AP_ComputerLab
jupyter$ cd USPAS_AP_ComputerLab
USPAS_AP_ComputerLab$ cp IntroPython.ipynb ../LabExercise/
USPAS_AP_ComputerLab$
```

- Work in LabExercise directory

Python & Jupyter Notebook Tutorial

- Open “IntroPython.ipynb”

References

- Aseev, V. N., Ostroumov, P. N., Lessner, E. S., & Mustapha, B. (2005, May). TRACK: The new beam dynamics code. In *Particle Accelerator Conference, 2005. PAC 2005. Proceedings of the* (pp. 2053-2055). IEEE.
- Franchi, A., Duperrier, R., Franchetti, G., Gerigk, F., Groening, L., Hofmann, I., ... & Yaramyschev, S. (2005, June). Benchmarking linac codes for the HIPPI project. In *AIP Conference Proceedings* (Vol. 773, No. 1, pp. 110-113). AIP.
- Shishlo, A., Cousineau, S., Holmes, J., & Gorlov, T. (2015). The particle accelerator simulation code PyORBIT. *Procedia Computer Science*, *51*, 1272-1281.
- Zhang, T., Liu, B., Chen, J., & Wang, D. (2016). Python-based high-level applications development for Shanghai soft X-ray free-electron laser.
- Cary, J. R., Abell, D. T., Bell, G. I., Cowan, B. M., King, J. R., Meiser, D., ... & Werner, G. R. (2016). Select advances in computational accelerator physics. IEEE Transactions on Nuclear Science, 63(2), 823-841.
- Rakitin, M. S., Chubar, O., Moeller, P., Nagler, R., & Bruhwiler, D. L. (2017, August). Sirepo: a web-based interface for physical optics simulations-its deployment and use at NSLS-II. In Advances in Computational Methods for X-Ray Optics IV (Vol. 10388, p. 103880R). International Society for Optics and Photonics.
- He, Z., Davidsaver, M., Fukushima, K., Maxwell, D., Shen, G., Zhang, Y., & Zhao, Q. (2017, May). Development Status of FRIB On-line Model Based Beam Commissioning Application. In *28th Linear Accelerator Conf.(LINAC'16), East Lansing, MI, USA, 25-30 September 2016* (pp. 100-103). JACOW, Geneva, Switzerland.