USPAS – *Simulation of Beam and Plasma Systems*

Steven M. Lund, Jean-Luc Vay, Remi Lehe, Daniel Winklehner and David L. Bruhwiler

Lecture: Graphical User Interfaces

Instructor: David L. Bruhwiler

Contributors: P. Moeller, R. Nagler and C. Hall

G. Andonian, UCLA / RadiaBeam Tech

U.S. Particle Accelerator School sponsored by **Old Dominion University**

http://uspas.fnal.gov/programs/2018/odu/courses/beam-plasma-systems.shtml

January 15-26, 2018 – Hampton, Virginia

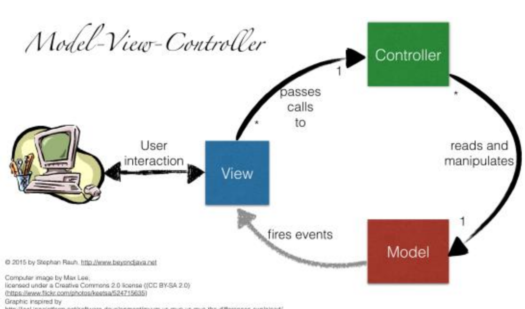
This material is based upon work supported by the U.S. Department of Energy, Office of Science, Offices of High Energy Physics and Basic Energy Sciences, under Award Number(s) DE-SC0011237 and DE-SC0011340.

Goals

- Graphical User Interfaces (GUI)
  - understand some principles of user interface (UI) design
  - appreciate the difficulties associated with desktop GUIs
  - consider some aspects of "software sustainability"
- Understand what's meant by "cloud computing"
  - Why this is helpful for "computational reproducibility"
  - Other benefits it can provide, like easy collaboration
- Learn a little about the elegant code from ANL
  - M. Borland, "elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation," Advanced Photon Source LS-287 (2000).
  - Y. Wang and M. Borland, "Pelegant: A Parallel Accelerator Simulation Code for Electron Generation and Tracking," AIP Conf. Proc. 877, 241 (2006).
  - https://ops.aps.anl.gov/manuals/elegant_latest/elegant.html
- Become familiar with Sirepo/elegant
  - a browser-based GUI

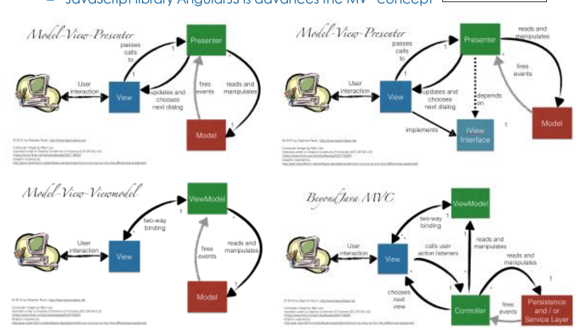
Separate `Physics` ***from*** `UI` ***from*** `Control logic`

- Commonly referred to as model-view-controller (MVC)



MVC has become **Model – View –** `Whatever`

- The reality of modern UI's is complicated
  - JavaScript library AngularJS is advances the MV* concept https://angularjs.org



Why do we end up with Cross-platform GUIs?

- GUI used to imply "desktop application"
  - still does in for some people
  - Windows-only vs Linux-only vs Linux & MacOS vs Mac-only
  - there is immediate frustration from users
    - **strong pressure to support multiple platforms**
- The Qt application and UI framework is a popular solution
  - cross-platform C/C++ GUI toolkit, http://www.qt.io
  - Python bindings, http://riverbankcomputing.com/software/pyqt
  - there are a number of competing open source options
- It's expensive to develop & maintain a cross-platform application
  - Qt / Python help a lot, but do not solve the problem
  - see slide #8 of the "computational reproducibility" lecture
    - **Python 2.7.x code is not always compatible with Python 3.x code**
    - **32 bit and 64 bit versions of Python are incompatible**
    - **open source library projects issue frequent releases**
    - **underlying physics application may not be robustly cross-platform**

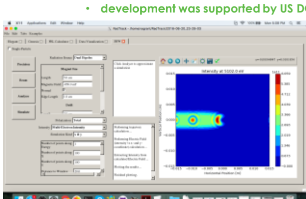
Why so few GUIs for particle accelerator codes?

- There are definitely some, but…
  - how many particle accelerator codes are there? `Many`
  - how many users are there for each code? `Not so many`
  - how many OS's are used by each subset? `Probably 3`
- Too expensive to support M GUIs on 3 platforms
  - only of order N_{total} / (3 * M) users for each instance
  - even if you get someone else to pay the cost, is it worth it?
  - question of software sustainability
- Also, code development teams are busy and under-funded
  - they will not modify their code to support GUI development efforts
  - any code/GUI coupling must be very loose
  - all burden is on the GUI developer to support file formats, etc.
- An approach was proposed and developed by RadiaBeam Tech
  - a Python/Qt cross-platform GUI for multiple physics codes
    - **very loose coupling between GUI and code**
    - **GUI enables easy interchange between different codes**
  - the project is called RadTrack

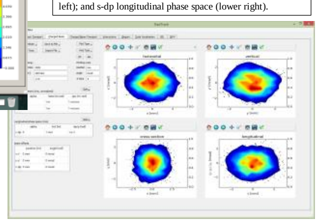
D.L. Bruhwiler, R. Nagler, S.D. Webb, G. Andonian, M.A. Harrison, S. Seung, T. Shaftan and P. Moeller, "Cross-platform and cloud-based access to multiple particle accelerator codes via application containers," Proc. Int. Part. Accel. Conf., MOPMN009 (2015).

RadTrack – a cross-platform GUI for accelerator codes

- Available on GitHub, https://github.com/radiasoft/radtrack
  - good place to start if you're interested in PyQt, with lots of good code
  - but it's no longer supported – a question of software sustainability
    - **development was supported by US DOE/BES,** Award # DE-SC0006284



A Synchrotron Radiation Workshop (SRW) simulation showing the interference of synchrotron radiation from two nearby dipoles – an important e- beam diagnostic.



RadTrack simulation, showing 2D phase space projections plots: horizontal x-x' phase space (upper left); vertical y-y' phase space (upper right); x-y configuration space (lower left); and s-dp longitudinal phase space (lower right).

Class discussion:

- Any questions at this point?
- Have you used an accelerator physics code with a GUI?
  - If yes, how was the GUI helpful (or not)?
  - If no, have you ever wished there was a GUI for codes you use?
- Have you ever used a GUI-based code and been frustrated?
  - do you consider it a point of honor to work from the command line?
- What is meant by "software sustainability"?

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

Sirepo: in-browser technologies

- *HTML5 (including JavaScript, CSS3, SVG, etc.)*
  - *https://en.wikipedia.org/wiki/HTML5*
- *Bootstrap,* *http://getbootstrap.com*
  - *fundamental for cross-platform web applications*
- *AngularJS,* *https://angularjs.org*
  - *model–view–whatever (MV *) architecture, components*
- *D3.js,* *http://d3js.org*
  - *interactive plots, data-driven transformations*
- *Karma,* *http://karma-runner.github.io*
  - *testing framework for browser-based applications*
- *JSON,* *https://www.w3schools.com/js/js_json.asp*
  - *JavaScript Object Notation – lightweight data-interchange format*

Sirepo: server-side technologies

- *Docker* *https://www.docker.com*
  - *enables rapid deployment of applications to the cloud*
- *Flask* *http://flask.pocoo.org*
  - *lightweight framework for web development with Python*
- *Celery* *http://docs.celeryproject.org*
  - *task manager*
- *RabbitMQ* *https://www.rabbitmq.com*
  - *message broker*
- *Jinja* *http://jinja.pocoo.org/docs/dev*
  - *secure and widely used templating language for Python*
- *Werkzeug* *http://werkzeug.pocoo.org/docs/0.10*
  - *Python utility library, compliant with the WSGI standard*
- *Nginx* *https://www.nginx.com/resources/wiki*
  - *HTTP server & proxy; scalable event-driven architecture*
- *Pyenv* *https://github.com/yyuu/pyenv*
  - *Python version management, multiple versions*

Class discussion:

- Any questions at this point?
- How does cloud computing help with ease of use?
- How does cloud computing help with software sustainability?

Notable Applications of elegant

- Beam transport lines for the SSRL pre-injector
- All APS accelerators now use **elegant**-designed optics
  - Design of the APS Positron Accumulator Ring and transport lines
  - Low-emittance optics development for the APS and APS booster
  - Design of bunch compressor and new linac optics for LEUTL
- APS top-up safety tracking
  - Ushered in a new mode of storage ring operation
- LCLS start-to-end and S2E jitter simulations
  - Discovered CSR-driven microbunching instability
- Used world-wide for FEL driver linac design, e.g.,
  - SLAC, DESY, BESSY, Sincrotrone Trieste, SPRing-8
- Used world-wide for ERL projects, e.g.,
  - Cornell, JLAB, BNL, Daresbury, JAERI
- Used to design the LTI Compact Light Source (commercial accelerator).

Quality Control

- **elegant** is important to APS operations and to users around the world
- Quality control is taken very seriously
- Source code is in CVS for version control and tracking
- Use extensive regression testing to "guarantee" that program updates don't break anything
  - When a feature is added, it is thoroughly tested
  - Selected test results are saved and used for checking of later versions
  - Program design allows us to largely automate this process.

Computer Science Philosophy

- Adopt a tool approach
  - Graphics and display functions external to program
  - Vastly simplifies the simulation code
  - Allows common pre- and post-processing tools for many codes
- External scripting required and supported
  - Program delivers <u>data</u> to user, not graphs of data
  - Data provided in a form that emphasizes scripting, not manual examination
  - Simplifies the program while empowering the user
  - Allows use of open-source scripting languages
  - Well suited to automated and cluster computing.

SDDS

- SDDS = Self-Describing Data Sets
  - A file protocol for data storage
  - A toolkit of programs that transform such files
  - A set of libraries for working with such files
    - *Support for C/C++, Tcl/Tk, Java, MATLAB, Python, FORTRAN*
  - A central component of the APS control system
- Knowing SDDS is key to using **elegant** effectively
  - Pre- and post-processing
  - Graphical and text output
  - Linking of multiple simulations and codes
  - Cluster computing.

Why Use SDDS Files?

- Programs that use SDDS can be made robust and flexible
  - Check existence, data-type, units of data instead of crashing or doing an incorrect calculation
  - Respond appropriately to the data provided
    - *Exit and warn user if required data is missing, has unknown units, etc.*
    - *Supply defaults for missing data (e.g., old data set)*
- Existing data doesn't become obsolete when the program is upgraded
- Self describing files make generic toolkits possible, which saves effort on writing pre- and post-processors
- SDDS-compliant programs are "operators" that transform data
  - Using pipes allows concatenating operators to make a complex transformation
  - UNIX-like philosophy: everything is a (self-describing) file.

Toolkit Approach to Physics

- **elegant** is a very complex SDDS "operator."
  E.g., it
  - Transforms phase space from beginning to end of a system
  - Transforms magnet parameters into, e.g., Twiss parameters, tunes
- All input to and output from **elegant** is in SDDS files *except*
  - Lattice structure
  - Command stream
- **elegant**'s capabilities are augmented by other operators
  - SDDS toolkit: a large collection of inter-operative data analysis, manipulation, and display programs
  - SDDS-compliant physics programs.

Capabilities of elegant: elements

- **elegant** has no assumed "best" approach to modeling
  - User can often select from expedient methods
- Matrix methods
  - Second-order matrices for drifts, solenoids, bends, and correctors
  - Third-order matrices for quads, sextupoles, and alpha magnets
  - User-supplied second-order matrix
- Symplectic methods:
  - Fourth-order Ruth integrator for bends, quads, sextupoles, higher multipoles.
  - Hamiltonian has exact energy dependence
  - Can invoke classical synchrotron radiation and quantum excitation for tracking.

Capabilities of elegant: elements

- Time-dependent elements
  - Kicker with user-specified waveform
  - Rf cavities with exact phase dependence
    - *Simple cavity with perfect source*
    - *Phase-, frequency-, and amplitude-modulated*
    - *Phase-, frequency-, and amplitude-ramped*
    - *User-specified on-axis field profile*
    - *Deflecting cavity with noise and modulation*
  - Momentum ramp
  - Traveling-wave accelerator.

Capabilities of elegant: elements

- Numerically-integrated elements
  - Planar undulator with co-propagating laser beam
  - Solenoid from user-supplied field map
  - Dipole with extended fringe fields
- Apertures/material
  - One- and two-sided rectangular, elliptical, and super-elliptical collimators
  - Scrapers with optional elastic scattering
  - Foil elastic scattering.

Capabilities of elegant: elements

- Collective effects
  - Intra-beam scattering in rings (L. Emery)
  - Short-range longitudinal and transverse wakes
  - Longitudinal- and transverse rf modes
  - Coherent synchrotron radiation in dipoles and drifts
  - Longitudinal space charge in drifts and cavities
  - Linear transverse space charge (A. Xiao)
- Diagnostics
  - Beam position monitors
  - Particle coordinate/property analysis points (to SDDS file)
  - Histogram analysis points (to SDDS file).

Capabilities of elegant: elements

- Miscellaneous
  - SCRIPT element will incorporate an external program as an element in an elegant lattice
  - User-specified scattering distribution
  - Lumped-element synchrotron radiation
  - Pick-up/driver elements for simulating transverse single-bunch digital feedback in rings.

Capabilities of elegant: Calculations

- Twiss parameter computation
  - Optionally-computed on-orbit and with errors
  - Includes radiation integrals
  - Includes chromaticity to third order and first-order tune shift with amplitude
  - Optional computation of coupled Twiss parameters (V. Sajaev)
  - SDDS output
- Transport matrix
  - Optionally computed on-orbit and with errors
  - SDDS and text output vs s (first- and second-order)
- Floor coordinates
  - Fully three-dimensional computation
  - SDDS output.

Capabilities of elegant: Calculations

- Variation
  - Nested sweeps of "any" parameters of any elements
  - Sweep using values supplied in an external SDDS file
- Errors
  - Errors for "any" parameter of any element
  - User-selectable distributions, amplitudes, cutoff, ...
  - Linking of errors between elements
  - Multiple error sets in one run
  - Loading of error sets from SDDS files
  - Saving of error values to SDDS files.

Capabilities of elegant: Calculations

- Saving and loading
  - Optimized lattice can be saved
    - *To a new (text) lattice file*
    - *To an SDDS lattice parameter file*
  - SDDS file can be manipulated with Toolkit, reloaded
  - SDDS file can be generated by another program to provide loadable custom error sets.

Capabilities of elegant: Calculations

- Closed orbit
  - Variable or fixed path length
  - On- and off-momentum
  - SDDS output
- Correction
  - Will correct tunes, chromaticities, and trajectory/orbit
  - Does these sequentially with optional iteration
  - SDDS output
    - *trajectory/orbit correction matrix*
    - *corrector strengths*
    - *correction statistics*
    - *quadrupole strengths*
    - *sextupole strengths.*

Capabilities of elegant: Calculations

- Optimization
  - Optimizes user-supplied penalty function depending on almost any calculated quantity
    - *Final or interior beam parameters from tracking*
    - *Final or interior Twiss parameters*
    - *Global values like equilibrium emittance, tunes, chromaticities*
    - *Final or interior matrix elements*
    - *Final or interior floor coordinates*
  - Uses Simplex by default, but has other methods.

Capabilities of elegant: Calculations

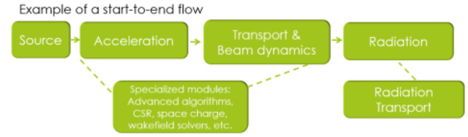
- Beam generation
  - Gaussian, hard-edge, and other distributions
  - Optional quiet-start sequences
  - Bunch train generation
  - Initial distribution can be saved to SDDS file
- Beam importation
  - Load beam from SDDS file
    - *Previously-generated and saved*
    - *Previously tracked*
  - Sequences of beams in one SDDS file.

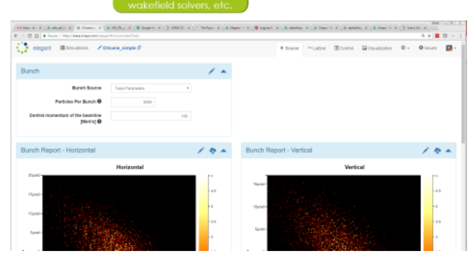
Capabilities of elegant: Calculations

- Aperture finding (dynamic and physical)
  - Searches for aperture boundary with optional subdivision of search interval
  - Various search modes
    - *Single- and multi-particle search starting from large amplitudes*
    - *Search along one or more lines starting at zero amplitude*
- Particle losses
  - Optional output of locations and coordinates of lost particles
  - Optional output of initial coordinates of transmitted particles.

The accelerator design workflow

Example of a start-to-end flow





Wrap up

- Any final questions regarding the material in this lecture?
- Have you ever used elegant?
  - If yes, then what application did you address?
  - If no, then do you think it could be useful to your work in the future?
- In the Computer Lab this afternoon, we will…
  - go through a demo of using Sirepo/elegant
- Acknowledgments
  - I borrowed several slides describing 'elegant' from a talk by M. Borland
  - M. Borland, "Introduction to Elegant," August 18, 2006 https://ops.aps.anl.gov/presentations/borland-2006-08-18.pdf
  - You can find the 'elegant' user manual here, https://ops.aps.anl.gov/manuals/elegant_latest/elegant.html