RadTrack User Manual

**Gerard Andonian, Mark Harrison and Stephen Seung**

RadiaBeam Technologies, LLC

1717 Stewart Street

Santa Monica, CA 90404-4021, USA

**Stephen Webb and David L. Bruhwiler**

RadiaSoft LLC

1348 Redwood Ave.

Boulder, CO 80304-1146, USA

**Timur Shaftan**

Brookhaven National Laboratory

Photon Sciences Directorate

Upton, NY 11973, USA

Contents

[A. Overview 2](#_Toc404241514)

[B. Description of the RadTrack tabs 2](#_Toc404241515)

[B.1. The Laser Mode Tab 3](#_Toc404241516)

[B.2. The Charged Particle Beam Tab 6](#_Toc404241517)

[B.3. Beamline Lattice Designer Tab 8](#_Toc404241518)

[B.4. Optical Beamline Design and Transport Tab 13](#_Toc404241519)

[B.5. The Code Interactions Tab 14](#_Toc404241520)

[B.6. The Data Visualization Tab 15](#_Toc404241521)

[B.7. The Free Electron Laser (FEL) Tab 22](#_Toc404241522)

[B.8. The Synchrotron Radiation Workshop (SRW) Tab 23](#_Toc404241523)

[B.9. The Elegant Tab 27](#_Toc404241524)

[C. Additional capabilities 29](#_Toc404241525)

[C.1. The Laser Heater Module 29](#_Toc404241526)

[C.2. The Relativistic Beam Wakefield Module 34](#_Toc404241527)

[D. Technical References 35](#_Toc404241528)

# Overview

**RadTrack motivation & philosophy**

Many light source applications and advanced accelerator concepts require the co-propagation and interaction of electron bunches with a laser or other coherent radiation field (e.g., laser-plasma accelerators, inverse photon scattering experiments, seeded free-electron lasers, femto-slicing experiments, and electron beam diagnostics with laser light). The different components of the experiment often require a different set of modeling tools, e.g. laser-electron beam interactions are typically studied on a case-by-case basis while FEL problems use a different dedicated simulation code. Laser-electron beam interactions and schemes compose a large subset of advanced acceleration topics and future x-ray radiation sources. Start-to-end modeling is required to completely resolve the fundamental underpinnings of these complex problems.

Many accelerator and radiation codes use complicated input files, with command-line or script-driven execution, a wide variety of data file formats, and the need for each user to develop post-processing and visualization. Use of two or more codes together for end-to-end design or for complicated sub-systems can be difficult and error prone. Unfortunately, it is not practical to develop a custom graphical user interface (GUI) for every community code, nor is it practical to develop custom conversion tools for every relevant combination of codes. Also, the development teams for these codes generally have little or no funding directed towards software development or maintenance, so the developers cannot make changes to accommodate 3rd party libraries or user interfaces.

RadiaBeam Technologies, LLC and RadiaSoft LLC are working together with the accelerator physics community to develop RadTrack – a light-weight GUI-driven framework for multiple codes, including automatic conversion between diverse file formats, designed to accommodate new versions of supported tools and future addition of other codes, without placing any burden on the code developers. At present, the RadTrack framework is used to build a cross-platform desktop application, based on Python [[[1]](#endnote-2)] and the Qt application and UI framework [[[2]](#endnote-3)]. The PySide toolkit [[[3]](#endnote-4)] is used to interface Qt’s C++ API with Python. Publication quality graphics and additional interactivity are provided via the Matplotlib library [[[4]](#endnote-5)], which renders plot windows inside the Qt widgets. At present, RadTrack provides a common interface for the electron tracking code Elegant [[[5]](#endnote-6),34,35] and two radiation codes: the Synchrotron Radiation Workshop (SRW) [[[6]](#endnote-7),29,30] and QUINDI [[[7]](#endnote-8),[[8]](#endnote-9)]. Support for additional codes is under development.

# Description of the RadTrack tabs

The RadTrack Python/Qt GUI is tabbed, with each tab providing unique capabilities. We describe these tabs in detail below.

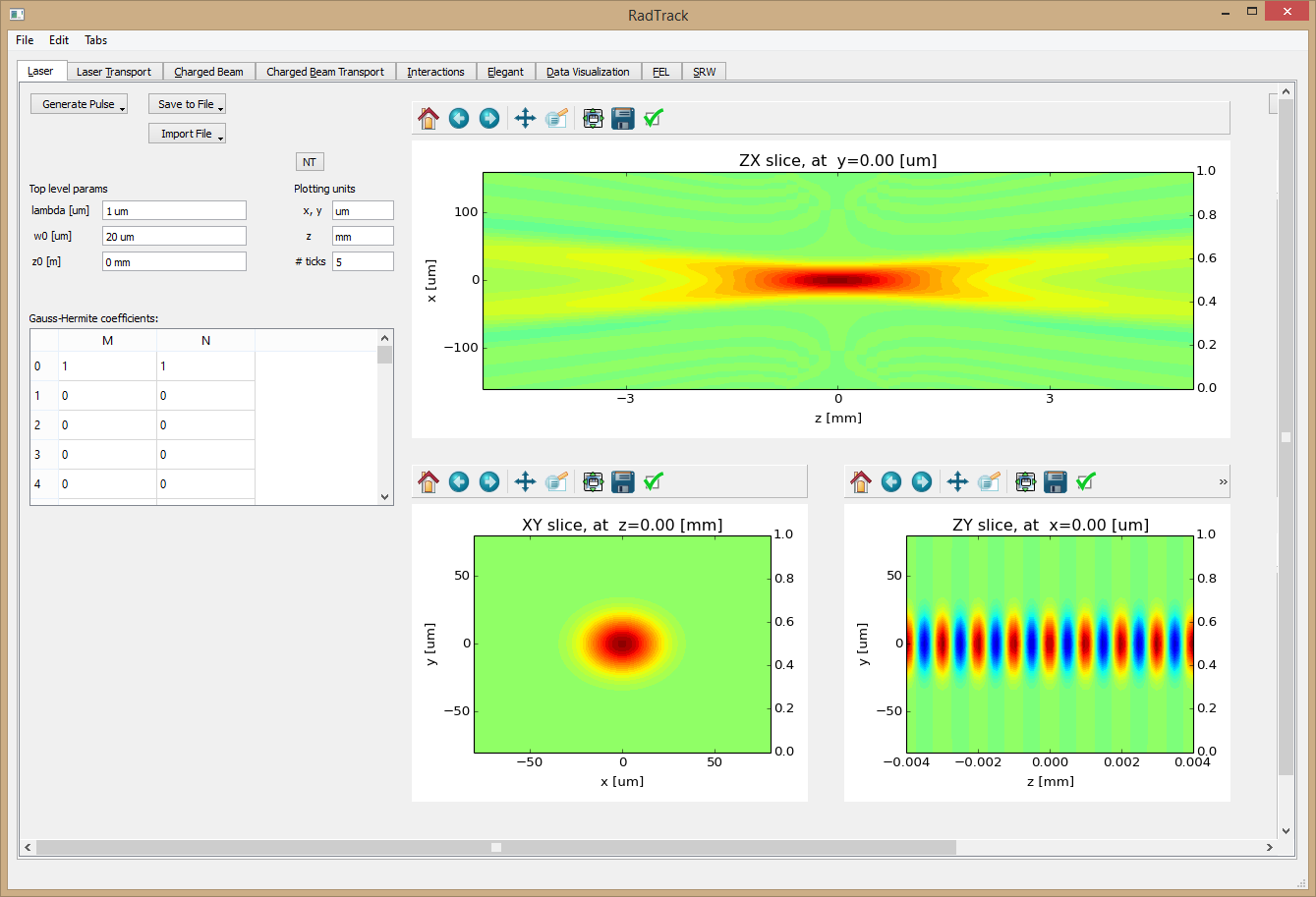
**Creating a tab to support a new code or capability**

Tabs are created using the Qt ‘creator’ [[[9]](#endnote-10)], which enables rapid design of a functional and intuitive GUI. A simple manual step is required to generate the PySide wrapping code, which enables the cross-platform interface between Qt and Python. The developer must then implement a Python class that defines all actions associated with the GUI elements (boxes, menu items, etc.).

Adding support for a specific code typically involves development of software to: 1) read/write some or all of the relevant data files; 2) execute the accelerator modeling application with appropriate command-line arguments, including hooks to a common API within RadTrack for managing paths, collecting error messages, browsing for files, etc.; 3) create input files for the code, based on GUI input specifications; 4) support non-standard data visualization and handling of special capabilities within the tab.

# The Laser Mode Tab

The laser mode tab is used to specify and visualize a long laser pulse in the paraxial approximation. The transverse profile is specified in terms of a standard Gauss-Hermite expansion. [[[10]](#endnote-11),[[11]](#endnote-12),[[12]](#endnote-13),[[13]](#endnote-14)] A screenshot is provided in Figure 1, showing the standard Gaussian mode.



**Figure 1**: Screen shot of the RadTrack ‘laser’ tab, showing a basic Gaussian laser mode. The upper plot shows color contours of the polarized E component (assumed to be aligned with the x-axis) in the x-z plane (intersecting the y-axis origin). The lower two plots show corresponding slices that intersect the other two axes.

Linear polarization is assumed, and the polarization direction can be arbitrarily specified. Additional polarizations will be supported in the future. Assuming polarization along the x-axis, the corresponding electric field component is written as follows:

 Eq. (1)

where and .

In Eq. (1) above, the transverse and vertical 1D Hermite expansions, *Hm* and *Hn* respectively, are implemented via the built-in SciPy [[[14]](#endnote-15)] function. Python’s complex arithmetic capabilities are used, enabling a direct connection with standard mathematical notation; however, the electric field values are returned as simply the real part of the final expression. The horizontal and vertical spot sizes, wx0 and wy0, can be specified independently, as can their corresponding locations, zx0 and zy0. As shown in Eqs. (2) below, the transverse coordinates can be arbitrarily shifted and rotated, and the longitudinal origin can also be arbitrarily shifted along its axis.

 Eq. (2a)

 Eq. (2b)

 Eq. (2c)

 Eq. (2d)

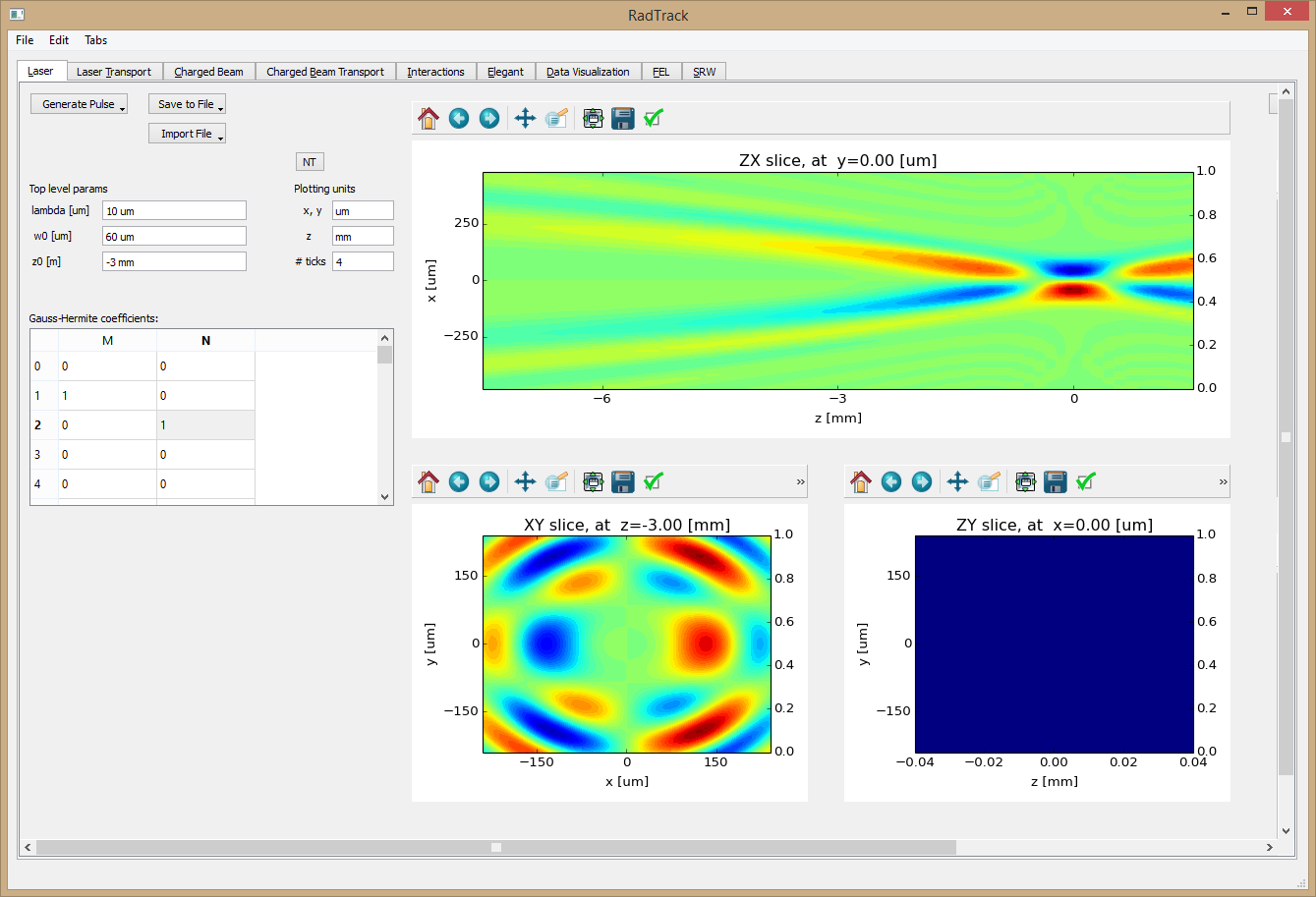
Figure 2 shows the laser tab with a (1-2) Gauss-Hermite laser mode. The central wavenumber is *k=2/*. The horizontal Rayleigh range, spot size and complex *q* factor are specified as follows:

 Eq. (3a)

 Eq. (3b)

 Eq. (3c)

The vertical equations are analogous. By default, the horizontal and vertical waists have the same size and longitudinal position.



**Figure 2**: RadTrack ‘laser’ tab; Gauss-Hermite laser mode (1-2). The lower-right panel is blank, because this mode has zero Ex values in the y-z plane that intersects the origin of the x-axis.

Our implementation of Gauss-Hermite modes builds on the standard high-performance python libraries *NumPy* [[[15]](#endnote-16)] and *SciPy* [14]. We used the complex syntax of standard optics textbooks, achieving the speed of compiled C code with the convenience, versatility and readability of Python. A few python classes encapsulate the logic in a logical and reusable way.

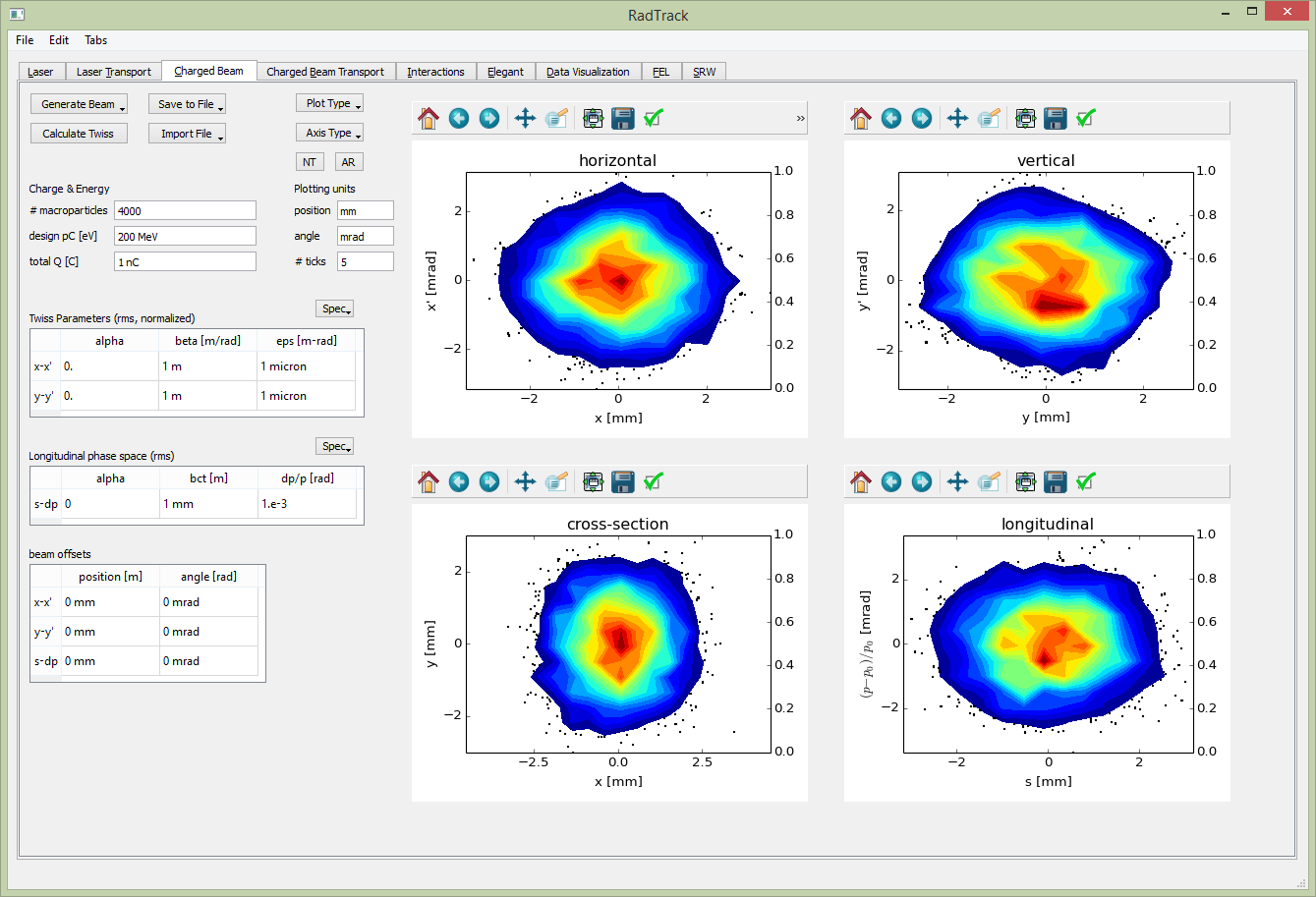
The built-in *SciPy* least squares fitting algorithm is used to fit a Gauss-Hermite expansion to arbitrary laser profiles that are provided by another design code or from experimental data. With the obtained fitting coefficients, a RadTrack user is able to simulate, for example, the resulting dynamics of laser heating for an electron beam (more details provided in the laser-heating section below). Figure 3 shows a relevant screenshot.



**Figure 3:** Screenshot of the RadTrack laser tab, showing Gauss-Hermite laser modes that approximate a mirror with a hole in it. The analytically specified transverse profile is shown in the upper-right. The Gauss-Hermite coefficients for one of the two expansions, obtained via least-squares fitting, are shown in the scrollable table on the left. The central plots and other parameters are as described above in Figure 1, but only one of the two expansions is shown. The blank plot in the lower-right will be used in the future to visualize differences (i.e. errors) between the desired profile and the resulting expansion.

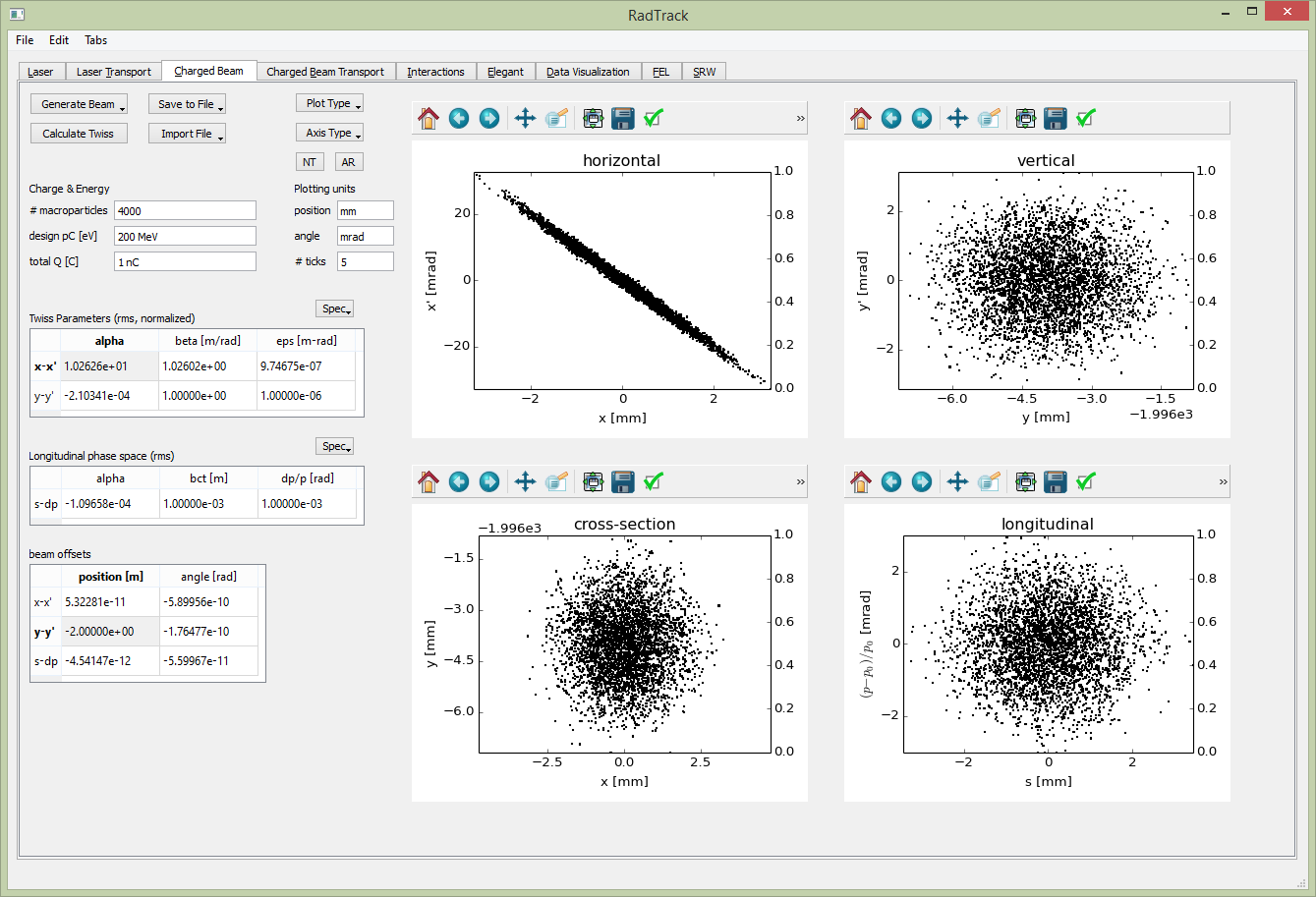
# The Charged Particle Beam Tab

The charged particle beam tab is used to specify and visualize a 6D particle distribution. The user chooses a distribution (only Gaussian is supported now, but several others will be added) and specifies the desired Twiss parameters [[[16]](#endnote-17)], misalignments, etc. Alternatively, an existing particle distribution can be read from a file (Elegant/SDDS format is supported, as well as an internal CSV format), for which the rms Twiss parameters will be automatically calculated and displayed for the user. User specified distributions can be written to a file for use by Elegant or other codes via the widely-used SDDS binary format [[[17]](#endnote-18),[[18]](#endnote-19),[[19]](#endnote-20),[[20]](#endnote-21)] or they can be used internally by other RadTrack tabs. A screenshot is provided in Figure 4.

**Figure 4**: Screen shot of the RadTrack ‘charged beam’ tab, showing four 2D phase space projection plots of the 6D particle distribution: 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). The combined contour/scatter visualization has been selected for this screen shot.

We implemented in Python a novel algorithm for particle beam generation, which takes as input standard rms Twiss parameters and emittances, as well as the desired number of macroparticles. First a spherical, Gaussian 6D phase space distribution is generated, with the RMS extent along each axis set to unity. Next the unwanted statistical correlations between all pairs of the six phase space axes are calculated, then subsequently removed via small-angle 6D rotations.

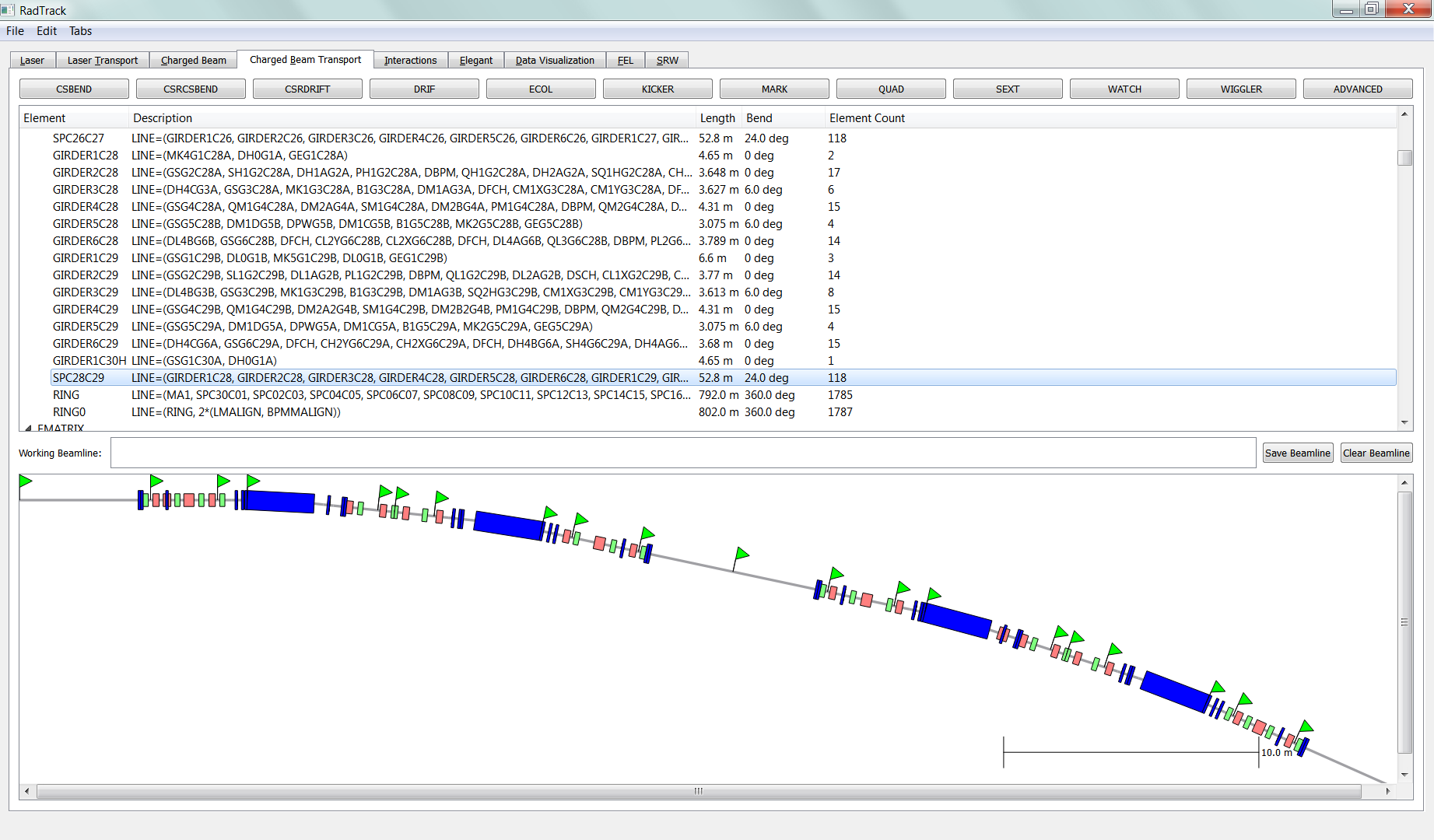
As a result, RadTrack produces the desired beam parameters with a minimum of statistical fluctuations. Statistical noise scales inversely with the square root of the particle number, making it numerically very expensive to decrease noise with more macro-particles. The implemented algorithm allows users to work with ~1,000 macro-particles, rapidly observing beam phase space evolution in accelerator lattices. Another screen shot of the tab is shown below in Figure 5.

**Figure 5:** Screen shot of the RadTrack ‘charged beam’ tab, analogous to Figure 4, but showing the scatter plots and using different beam parameters.

The charged particle beam tab supports three visualization options: scatter plots (seen in Figure 5), color contour plots, and combined scatter/contour plots (seen in Figure 4). Scatter plots over-emphasize low-density regions of phase space; while contour plots under-emphasize low-density regions and often miss them entirely. This is why the combined plots were developed.

# Beamline Lattice Designer Tab

The ‘charged beam transport’ tab enables RadTrack users to design, modify and visualize particle accelerator beamlines with an intuitive drag-and-drop interface. While choosing components and arranging them in beamlines, the user receives interactive visual feedback. A screen shot is provided below in Figure 6.



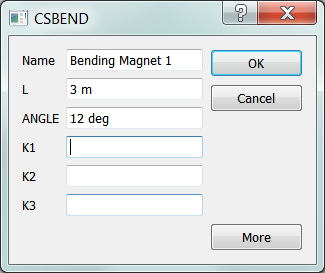
**Figure 6**: Screen shot of the ‘charged beam transport’ tab, showing a segment of a large storage ring. The mouse can be used to zoom in/out and drag the image.

The user begins constructing a beam line by creating individual beamline elements, or else by reading in an existing lattice file (Elegant format and a native RadTrack format are both supported). Common elements have their own buttons on the main screen, while more specialized elements like undulators, alpha magnets and deflecting cavities can be accessed with the far-right button labeled ‘ADVANCED’. These buttons can be seen in Figure 6 and are shown again in Figure 7. The beamline elements (names, parameters, etc.) are automatically extracted from the Elegant code documentation [42].



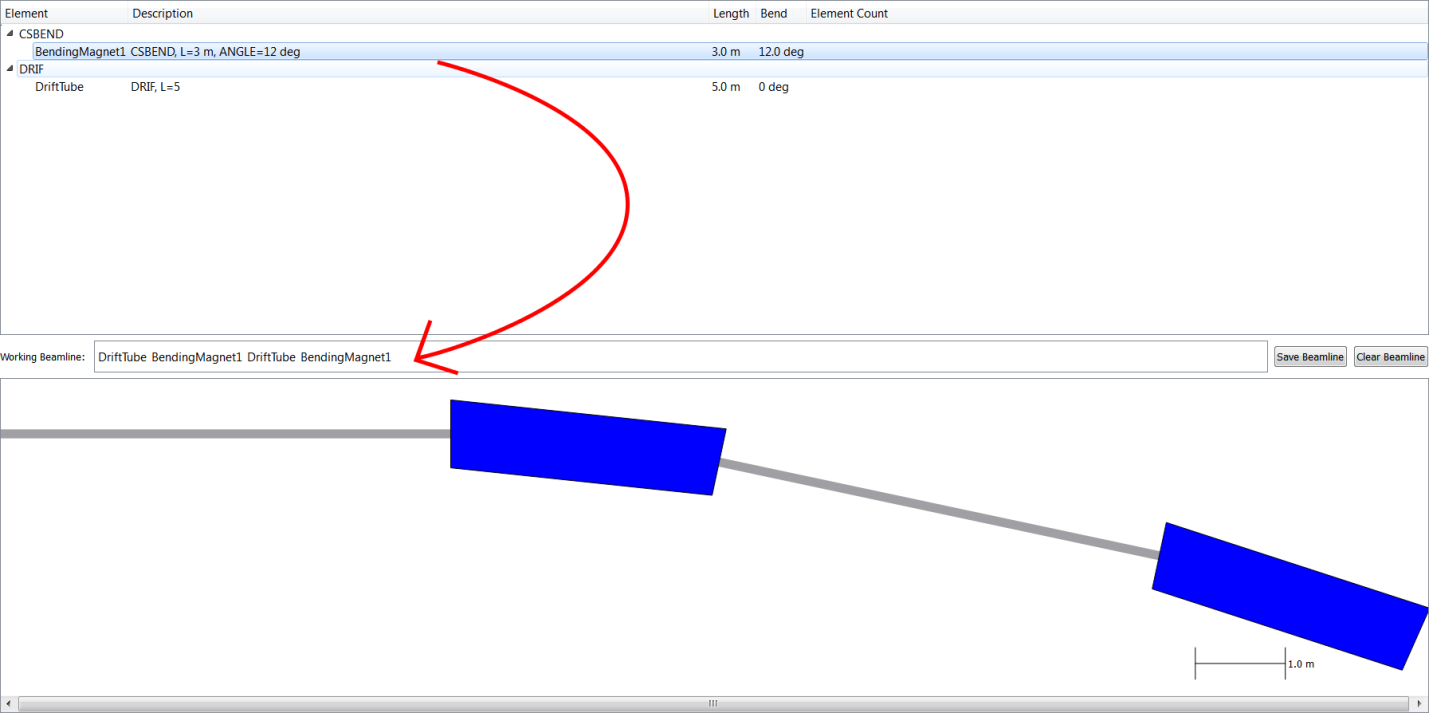
**Figure 7**: Beamline element creation buttons

When the user clicks on one of these buttons, e.g. ‘CSBEND’, a dialog box appears to define its properties. See Figure 8 for a screen shot. Next, the user can input the relevant parameters. More properties can be accessed through the ‘More’ button seen in the figure. Physical units may be entered together with the numerical parameter values. If the user selects a name for an element that is already in use or invalid in an Elegant source file, the new element will be automatically renamed to fix the problem.



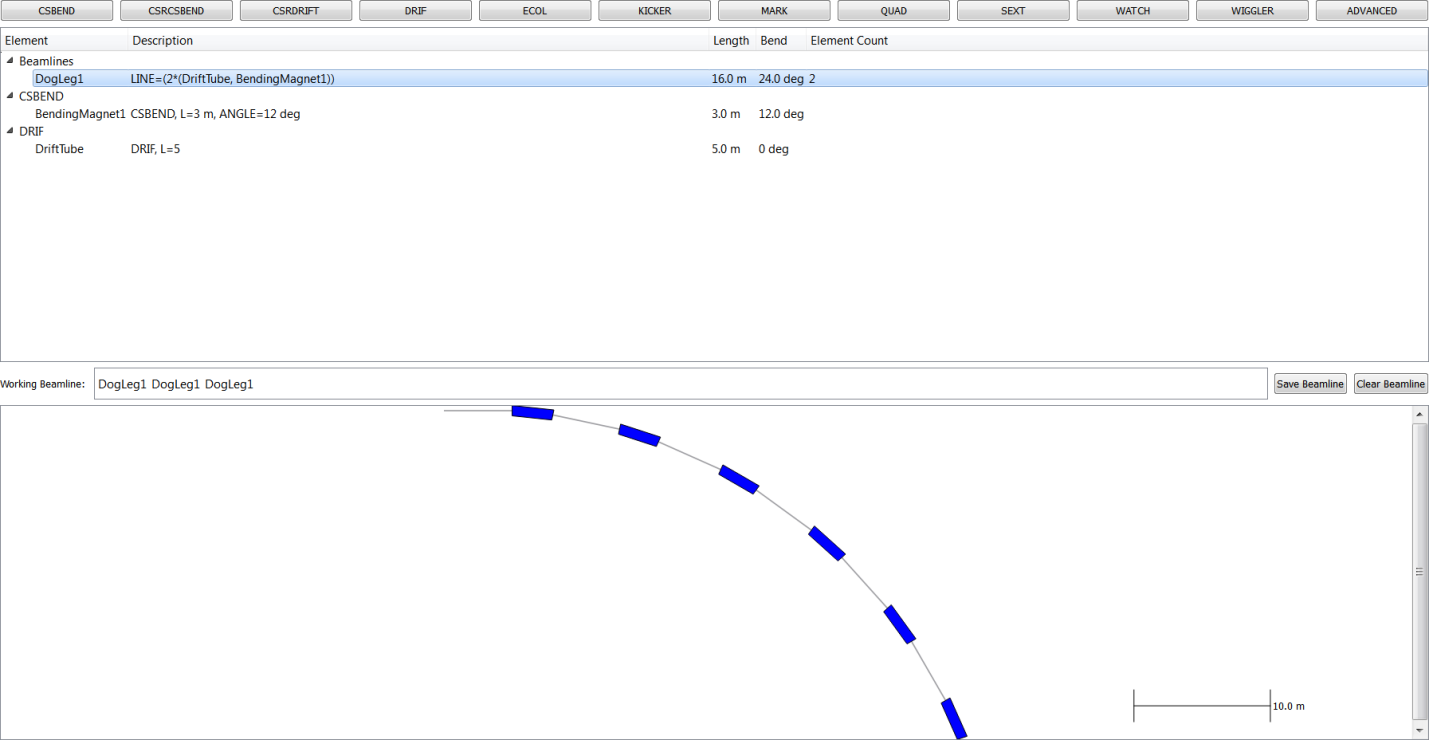
**Figure 8**: Properties dialog for a CSBEND bending magnet

Clicking the ‘OK’ button seen in Figure 8 adds the new element to the element list so that it may be used in creating new beamlines. The full Elegant-style specification is listed along with total length and bend. Beamlines have the further specification of the number of non-drift elements.



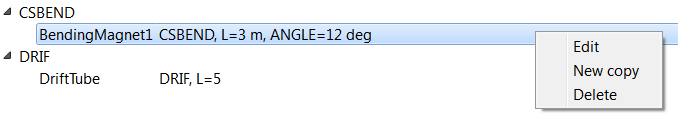
**Figure 9**: Screen shot of the ‘charged beam transport’ tab, showing the beamline element list and working beamline creation area. The arrow indicates how to click-and-drag elements to form a new beamline.

To create a new beamline, the user drags elements in the element list to the ‘Working Beamline’ creation area, as seen in the screen shot of Figure 9. Once the user is satisfied with a beamline section, clicking ‘Save Beamline’ adds it to the element list so that it can be used in creating larger beamlines, as shown in Figure 10. Clicking ‘Clear Beamline’ clears the Working Beamline area in case the user wants to start over. A length scale is shown in the lower-right corner of the graphical beamline preview area. The user can zoom in and out using the mouse wheel or the keyboard shortcuts ‘Ctrl +’ (zoom in) and ‘Ctrl -’ (zoom out).



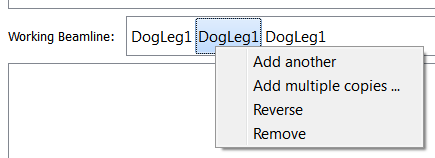
**Figure 10**: Screen shot of the ‘charged beam transport’ tab, showing an element list with a saved beamline and another beamline constructed from the first.

Right-clicking on various parts of the tab pops up context menus for editing elements and beamlines. For example, one can right-click on any item in an element list, as shown in Figure 11.

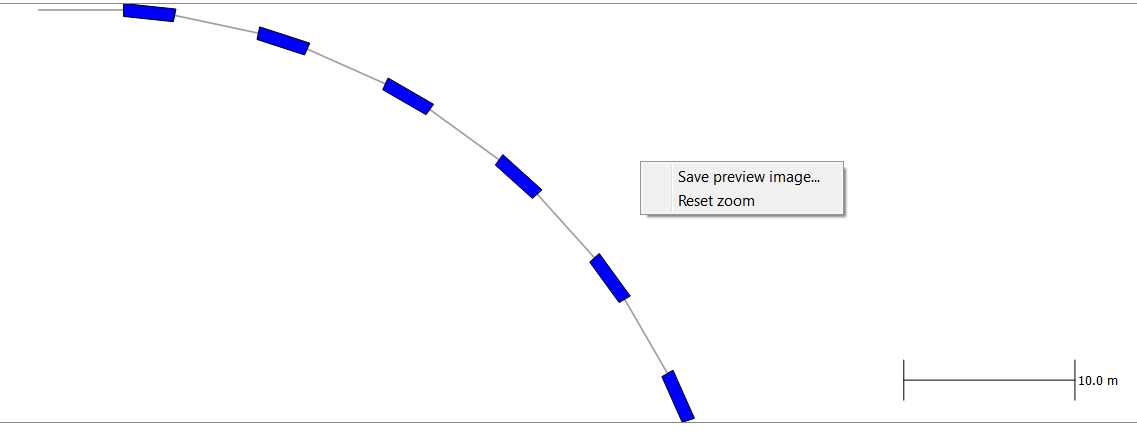


**Figure 11**: Screen shot of the context pop-up menu for element lists.

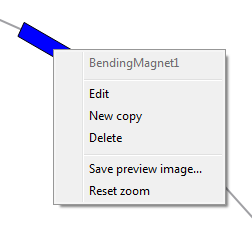
Similarly, one can right-click on any element in the working beamline, as shown in Figure 12. Selection of the ‘Reverse’ menu item creates a beamline with the elements in reverse order, the contained beamlines reversed, but without reversing the orientation of non-beamline elements. Likewise, one can right-click on the blank area in the graphical preview, as shown in Figure 13. If one right-clicks on a picture of a beamline element, a menu similar to Figure 14 will appear, identifying the element and combining the functionality of the element list and the graphical preview.



**Figure 12**: Screen shot of the context pop-up menu for the ‘Working Beamline’.

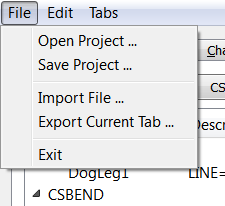


**Figure 13**: Screen shot of the context pop-up menu for the graphical preview of the lattice.



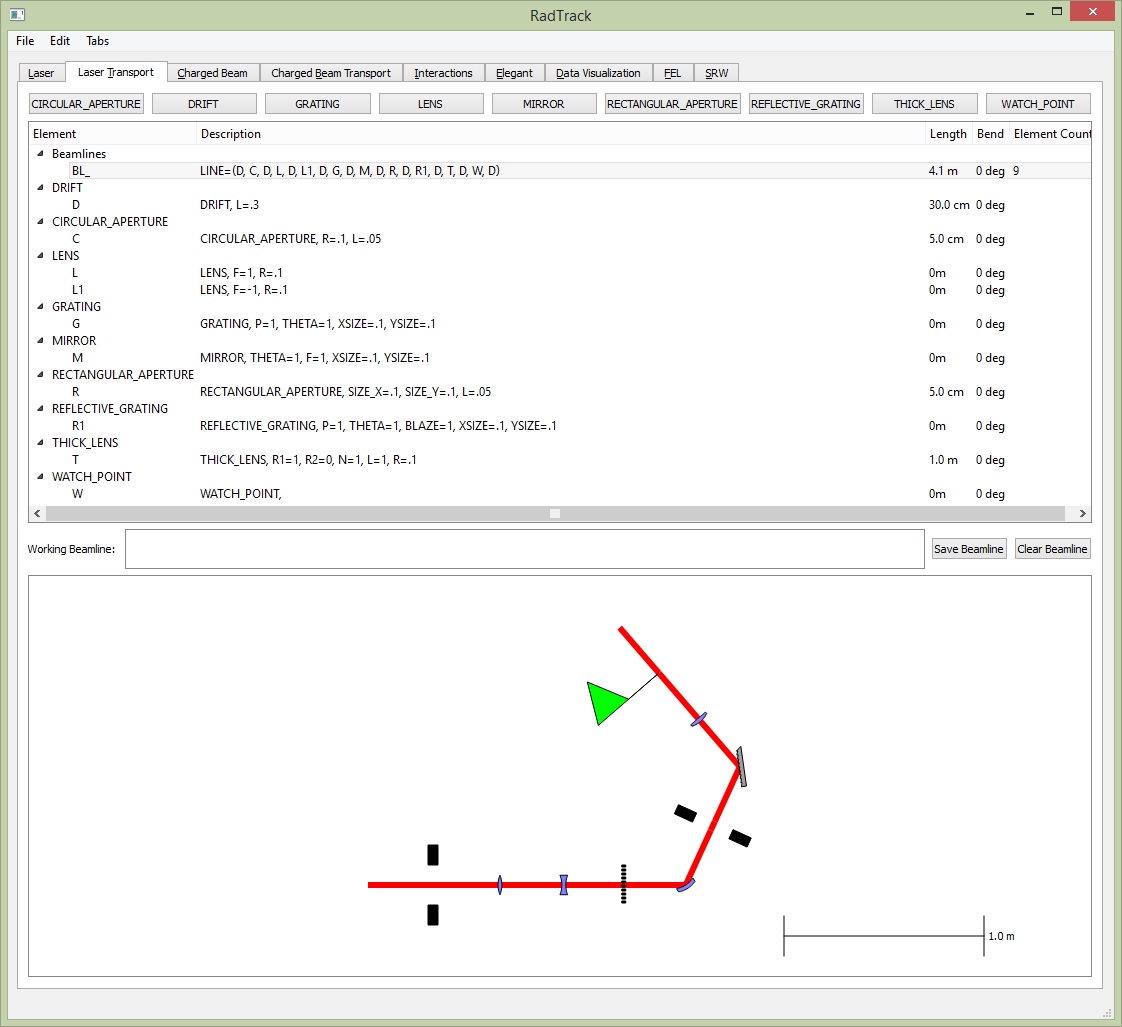
**Figure 14**: Right-click menu for element icons in the graphical preview area.

Under the ‘File menu’ (see screen shot in Figure 14), the user can select ‘Export Current Tab ...’ to export the created elements and beamlines into a file that can be read by Elegant. One can also import Elegant LTE files by selecting ‘Import File ...’. Name collisions with existing elements are handled by renaming the imported elements with appended numbers.



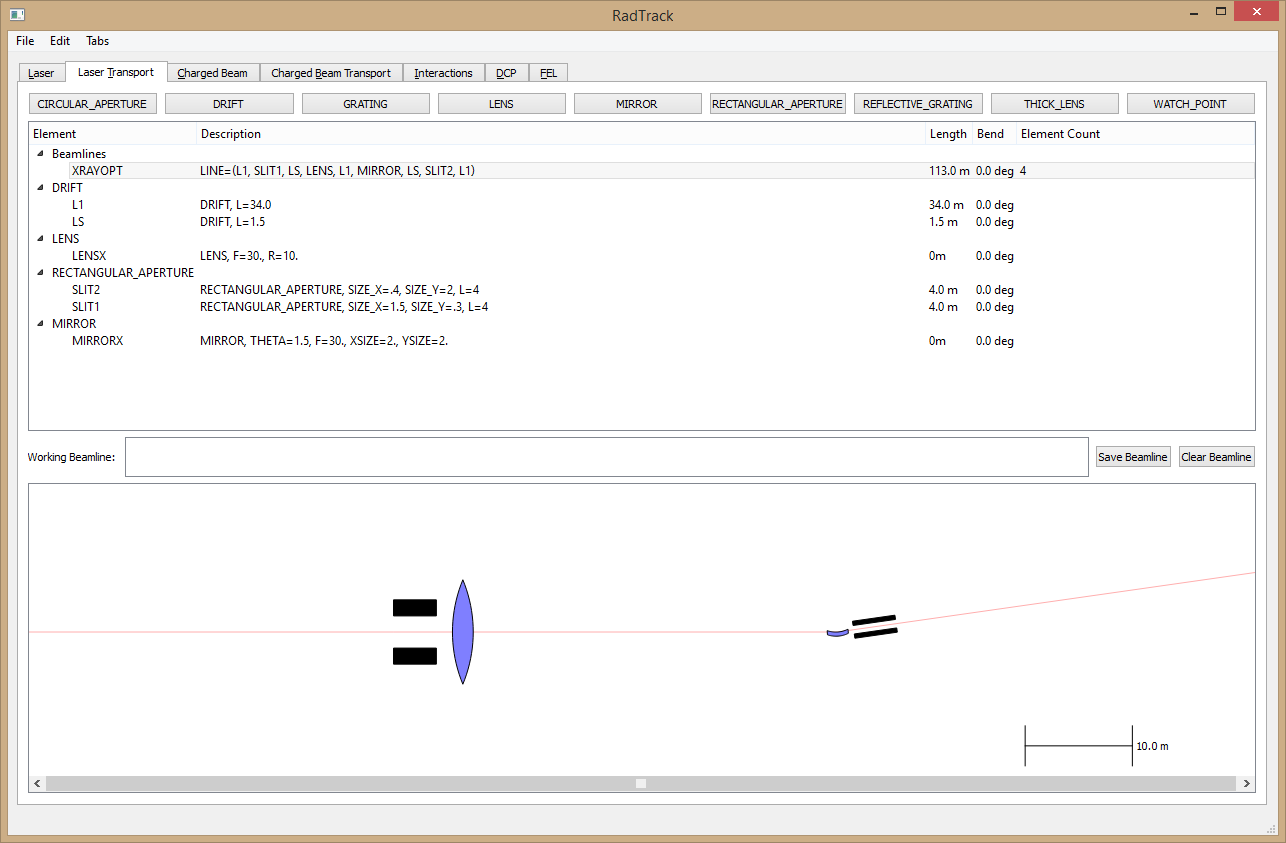
**Figure 15**: Screen shot of the top-level RadTrack File menu.

# Optical Beamline Design and Transport Tab

The ‘laser transport’ tab enables RadTrack users to design, modify and visualize optical beamlines with an intuitive drag-and-drop interface. While choosing optical components and arranging them in beamlines, the user receives interactive visual feedback. A screen shot is provided below in Figure 15.

**Figure 16**: Screen shot of the ‘laser transport’ tab, showing a sample of the available optical elements. The mouse can be used to zoom in/out and drag the image.

The drag-and-drop features and context pop-up menus described in the previous section “Beamline Lattice Designer Tab” are closely analogous to the features available here, so we do not repeat the discussion. Another screen shot is provided below in Figure 16, showing elements more typical of an X-ray transport line.

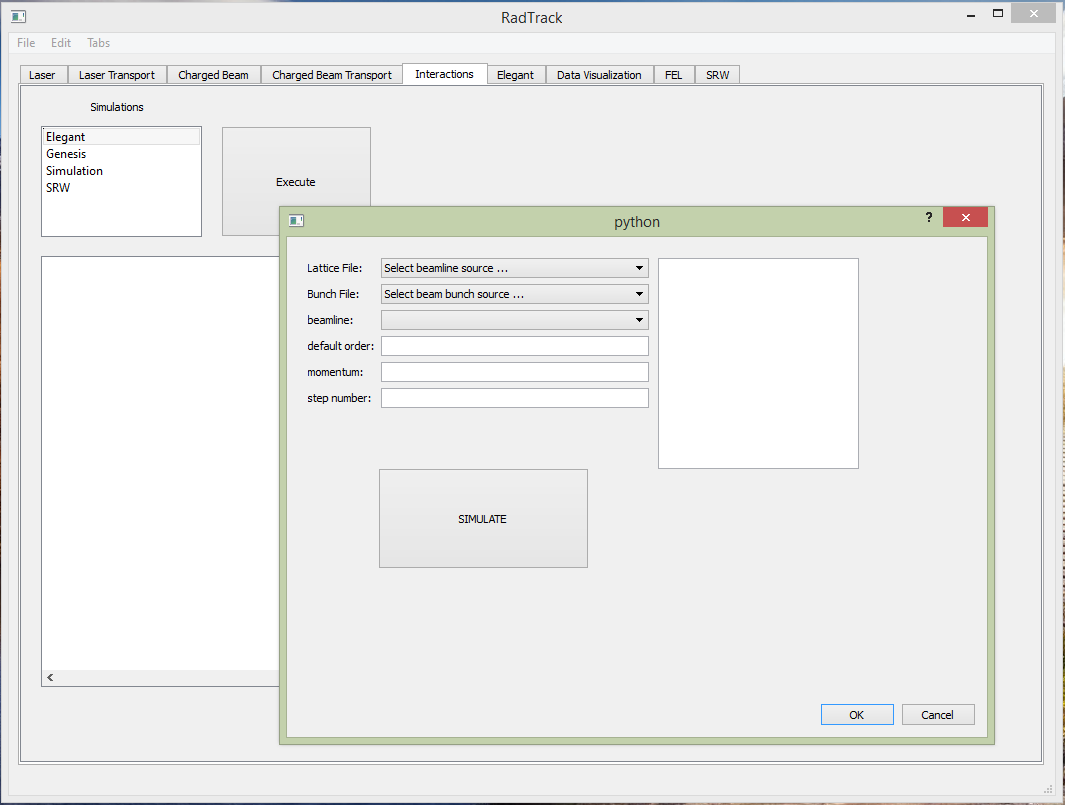
**Figure 17:** Screen shot of the ‘laser transport’ tab, showing slits and grazing incidence mirrors characteristic of X-ray optics. This image is adapted from Figure 1 of Ref. [[[21]](#endnote-22)].

# The Code Interactions Tab

The ‘interactions’ tab enables RadTrack users to work with multiple codes at once. Currently it supports simulating electron beam tracking with Elegant, followed by synchrotron radiation emission in an undulator, modeled by the Synchrotron Radiation Workshop (SRW) [6]. Users need only to drag-and-drop the desired simulation into the setup area on the bottom. A screen shot is provided in Figure 17.

When the selected code icon is ‘dropped’, a pop up window appears so the user can specify details of the desired simulation. This dialog is the same as the corresponding dialogs that user encounter in the Elegant and SRW tabs (see details in the sections below). In this case, however, selection of the ‘OK’ button near the bottom of the dialog window does not immediately execute the code – instead, it saves the simulation to be executed later.

A labeled green box then appears, representing the specified simulation. Dragging additional code icons onto the simulation box, generates a connecting line to indicate that output from the first simulation is used as input for the next simulation. For example, one can drag an ‘SRW’ icon onto a previously-dragged ‘Elegant’ icon. In the resulting simulation, the 6D electron macro-particle distribution, output from Elegant, is translated into an appropriate input for SRW. The user can then select the ‘Execute’ button to start the simulation.

****

**Figure 18**: Screen shot of the ‘Interactions’ tab, showing the pop up dialog that is generated when an ‘Elegant’ icon is dragged-and-dropped to the simulation panel.

# The Data Visualization Tab

The ‘Data Visualization’ tab enables RadTrack users to conveniently read and view data. This tab interacts with a sophisticated Python module that we call the Data Converter/Plotter (DCP). The DCP module serves as a translator for I/O data sets used in RadTrack and some common accelerator and synchrotron/laser radiation codes. At this point, Elegant/SDDS and SRW formats are supported. A screen shot is provided below in Figure 18.

**Figure 19**: Screen shot of the ‘Data Visualization’ tab, showing a particle distribution that was read in from an SDDS binary file generated by an Elegant simulation.

The DCP module stores data from multiple formats into internal arrays referred to as ‘parameters’ or data ‘columns’. Users can export data from the visualization tab as flat text files or in any of the supported code formats. In any case, the files can be documented with comments.

The data visualization tab allows the RadTrack user to plot columns of data in arbitrary combinations, and to analyze the data in various standard ways. The resulting plots are publication quality and can be copied into Microsoft Word or PowerPoint. While special purpose visualizations are handled by the corresponding tab(s), the data visualization tab enables the review and analysis of data from almost any file generated by the simulation. This allows users to get a quick understanding of the data, without requiring any special scripts or outside codes.

There are typically four types of data in the SDDS and SRW files parsed by the DCP module: legend, parameters, columns and variables. Here are some concrete examples of the four types:

1. **Legend**: text description of the file content
2. SDDS: &description text="Twiss parameters--input: bst-v8t.ele lattice: bst-v8.lte", contents="Twiss parameters", &end
3. SRW: #Bx [T], By [T], Bz [T] on 3D mesh: inmost loop vs X (horizontal transverse position), outmost loop vs Z (longitudinal position)
4. **Parameters**: single-value data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 4 values | | | |
|  | **Description** | **Name** | **Value** | **Units** |
| Npar values | Damping time |  | 5.30E-03 | s |
| Chromaticity |  | 133 |  |
| Radiated power | P | 6.45E+05 | W |
| …x.parDefinition | …x.parName[1] | …x.parData | … x.parName[3] |
| Momentum compaction | a | 8.33E-03 |  |

**Figure 20**: Table of ‘parameters’ data, as supported by the DCP module.

1. **Columns**: single-vector data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Ncol values | | | | |
| NColVal+3 values | **Description** | Beta function | Dispersion | x.colDef[3] | … | Energy |
| **Name** | y | x | x.colName | … | E |
| **Units** | m | m | x.colDef[2] | … | eV |
|  | 22.34 | 0.53 | x.colData[] | … | 1.00E+08 |
|  | 21.89 | 0.49 | x.colData[] | … | 1.00E+08 |
|  | 20.65 | 0.45 | x.colData[] | … | 1.00E+08 |
|  | … | … | … | … | … |
|  | 12.43 | -0.23 | x.colData[] | … | 1.03E+08 |

**Figure 21**: Table of ‘columns’ data, as supported by the DCP module.

1. **Variables:**
2. x.description SDDS/SRW Legend
3. x.parameterDefinition SDDS/SRW Parameter description
4. x.parameterName SDDS Parameter name
5. x.parameterData SDDS/SRW Parameter value
6. x.columnDefinition SDDS Column description
7. x.columnName SDDS Column name
8. x.columnData SDDS/SRW Column values

*Supported File Formats*

The DCP module serves as a translator for I/O data sets used in RadTrack and some common accelerator and synchrotron/laser radiation codes. The following formats are supported now (more will be supported in the future):

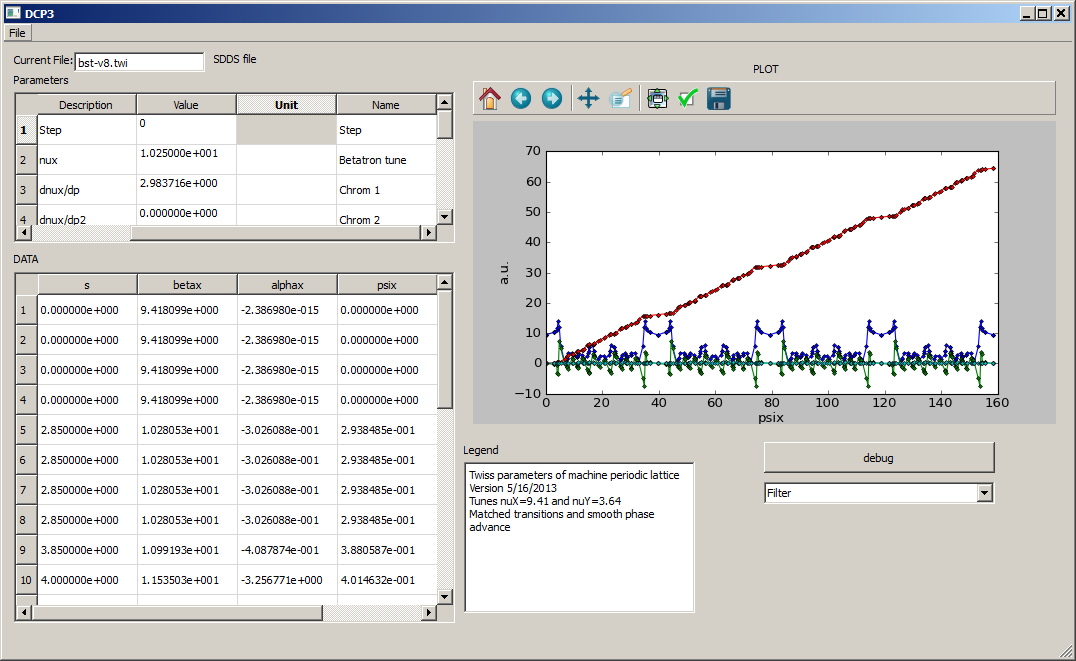
1. Elegant data in SDDS format
2. Quindi [8] output in HDF5 format
3. SRW output in text files
4. generic output in CSV format (e.g. export from MS Excel)

Output files generated by these codes contain data regarding beam phase space, particle beam distribution statistics, simulation parameters, Twiss parameters and other computed properties of the accelerator lattice, and other significant information. Tailored for accelerator scientists, the definitions of lattice, magnets, RF devices, and light propagation optics are included in the DCP interface.

*Guide to using the Data Visualization Tab*

To view data files, users select the following menu sequence: ‘File’ 🡪 ‘Import File’. Users then navigate to the desired file and open. The four types of data described above are loaded into corresponding tables. To plot the data, users then select the desired columns from a drop down menu above each column. Users may select only one column for the horizontal axis, but may select up to two for the vertical axis, with blue data points corresponding to the left vertical axis and the red data points corresponding to the right vertical axis. After selecting the desired columns, one need only select the ‘Plot’ button.

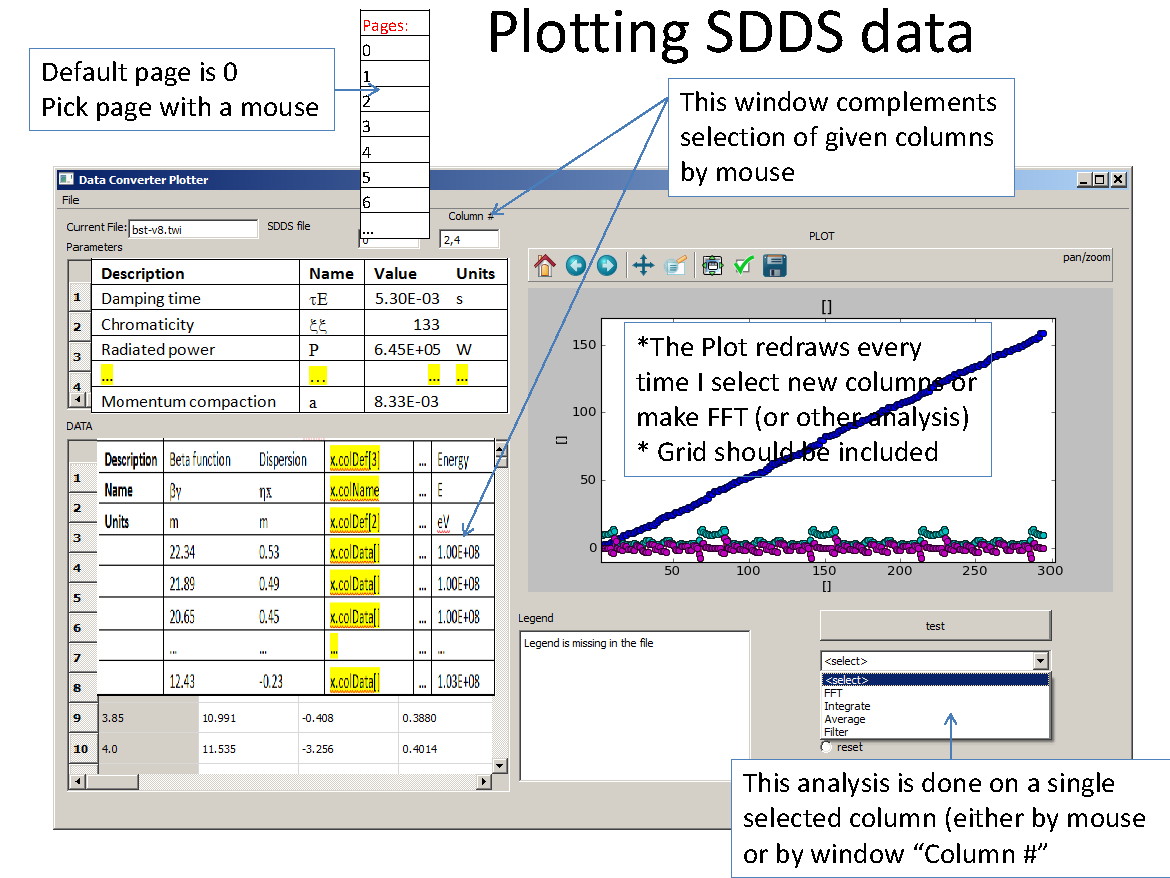
There’s also a quick view option, where users can simply select the column and a pop-up visualization window shows the general shape of the data. Users can also perform quick analysis on the column of data, such as averaging and fast fourier transforms (FFT). Some additional examples of usage are shown in the six screen shots (some of them annotated) on the next three pages: Figure 21, Figure 22, Figure 23, Figure 24, Figure 25 and Figure 26.

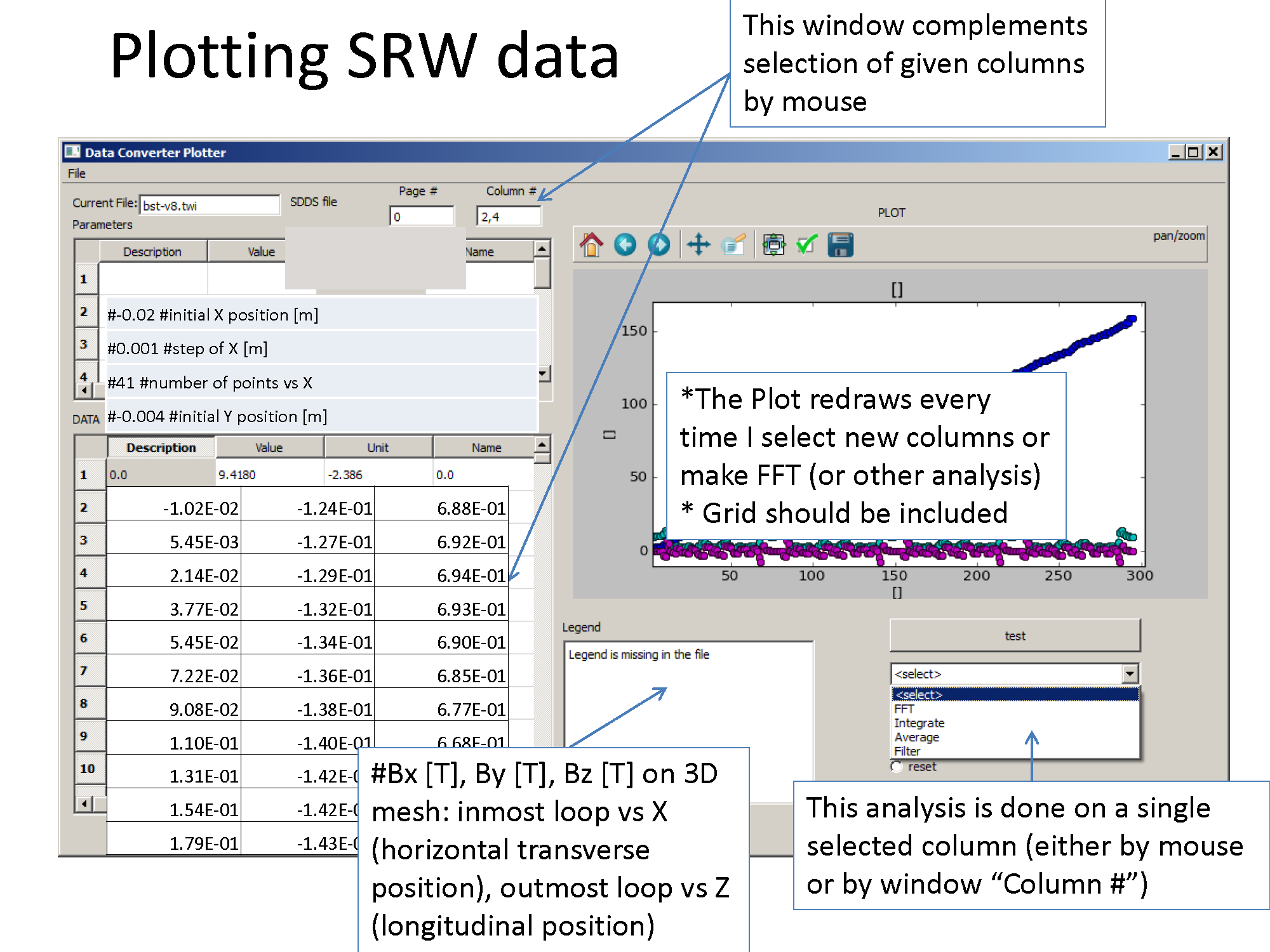


**Figure 22:**  Screen shot of the ‘Data Visualization’ tab, after reading in an SDDS file (a .twi file generated by Elegant). The plotting is straightforward as the user simply clicks the desired columns in the left window. The graphical output can be formatted (add labels, change colors, add legend, etc.) and exported as vector or bitmap graphical output.

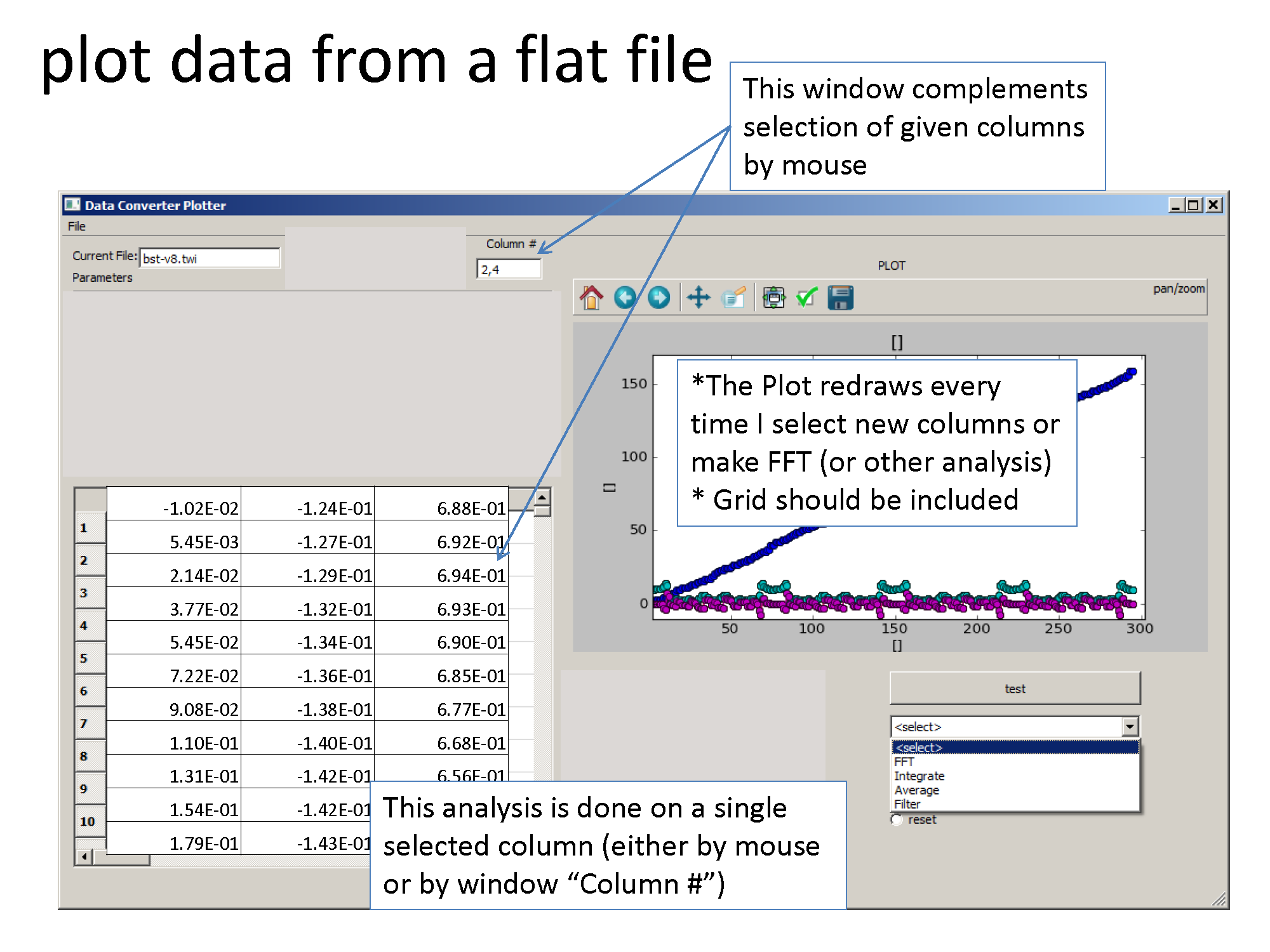
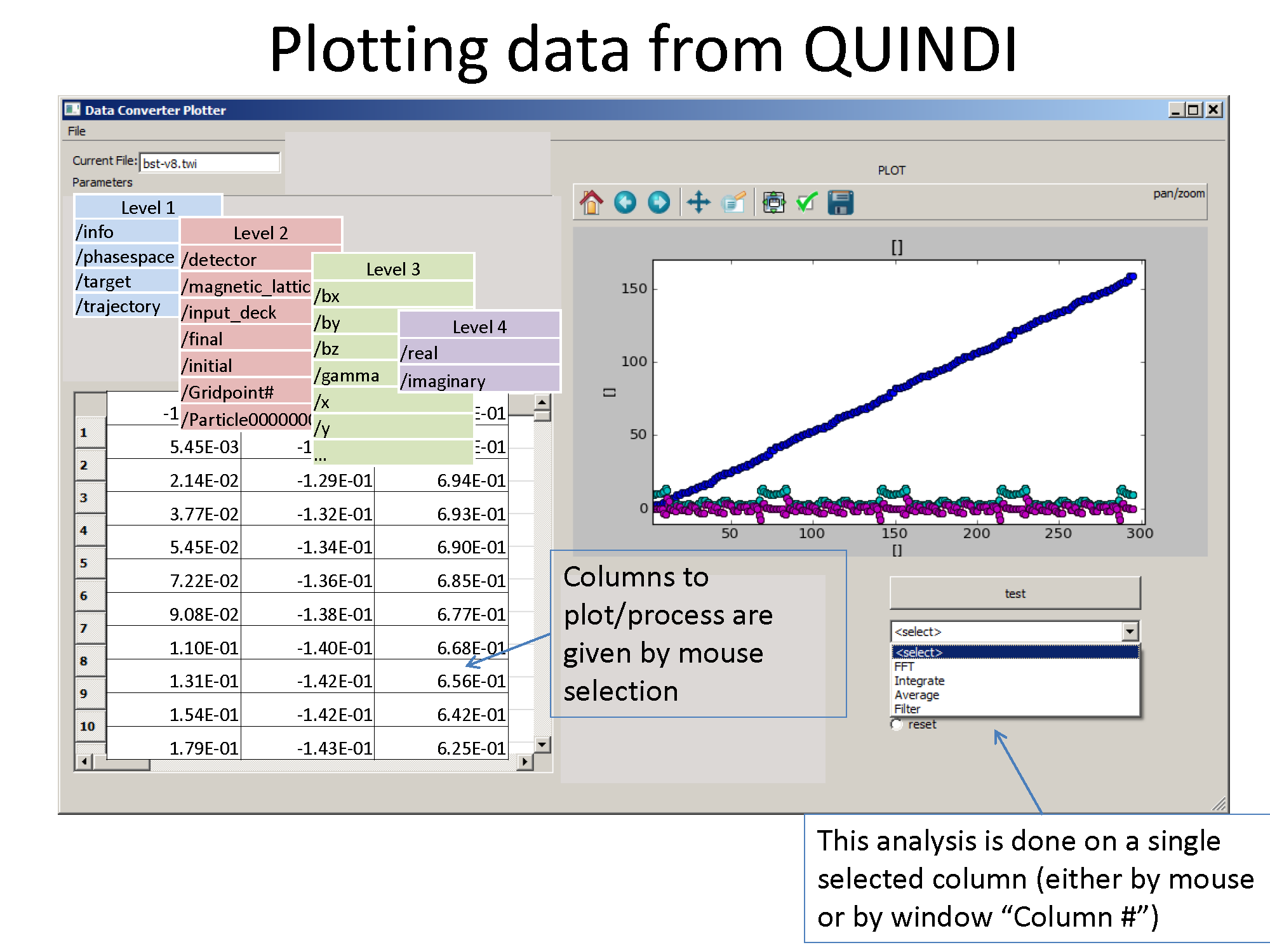


**Figure 23:** Screen shot of the ‘Data Visualization’ tab, after reading a data file generated by the SRW code.

**Figure 24:** Screen shot of the ‘Data Visualization’ tab, after reading an SDDS data file, with annotations to clarify usage and capabilities.

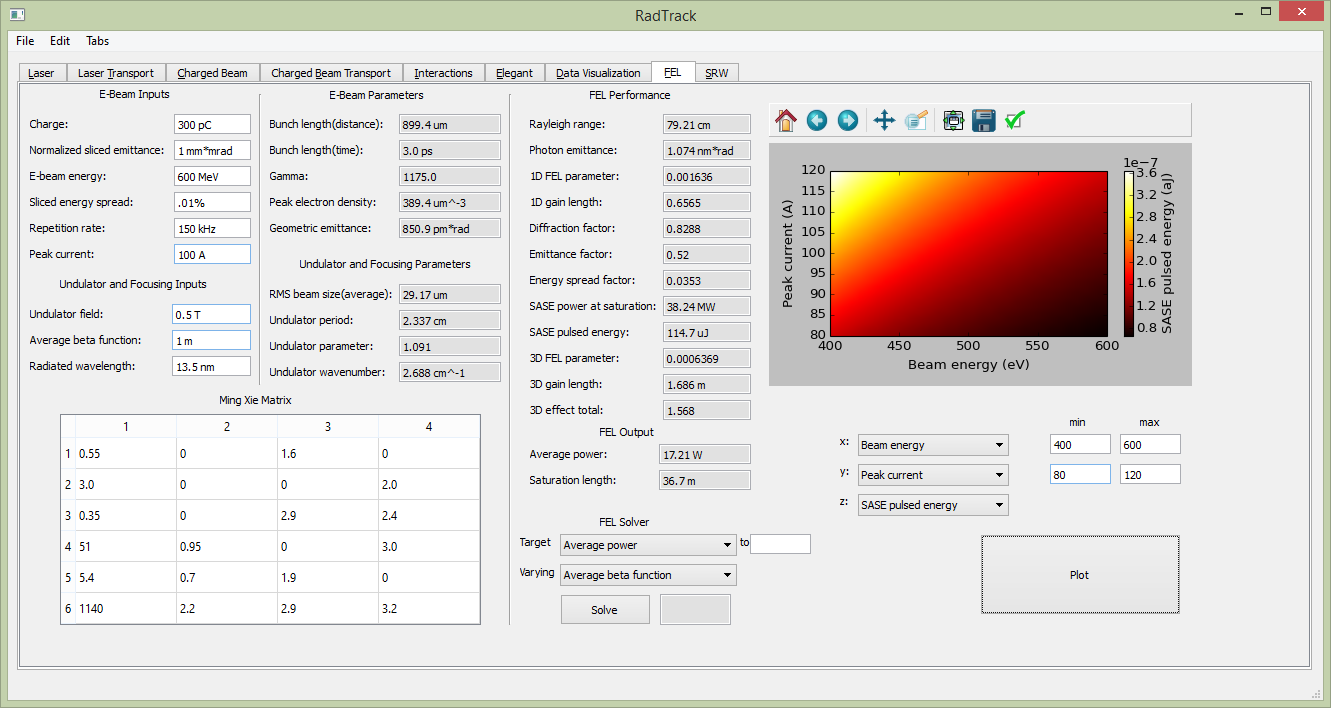
****

**Figure 25:** Screen shot of the ‘Data Visualization’ tab, after reading an SRW data file, with annotations to clarify usage and capabilities.

**Figure 26:** Screen shot of the ‘Data Visualization’ tab, after reading a generic CSV (comma sepatrated values) file, exported from MS Excel, with annotations to clarify usage and capabilities.

**Figure 27:** Screen shot of the ‘Data Visualization’ tab, after reading a QUINDI data file, with annotations to clarify usage and capabilities.

# The Free Electron Laser (FEL) Tab

The FEL tab provides rapid estimates of FEL performance, based on specified electron beam and undulator parameters. The calculation is based on the universal scaling function of Xie [[[22]](#endnote-23)]. The solution is given in the form of a polynomial with 19 fitting parameters listed in the table in the lower-left corner of the tab (see Figure 27 screen shot).

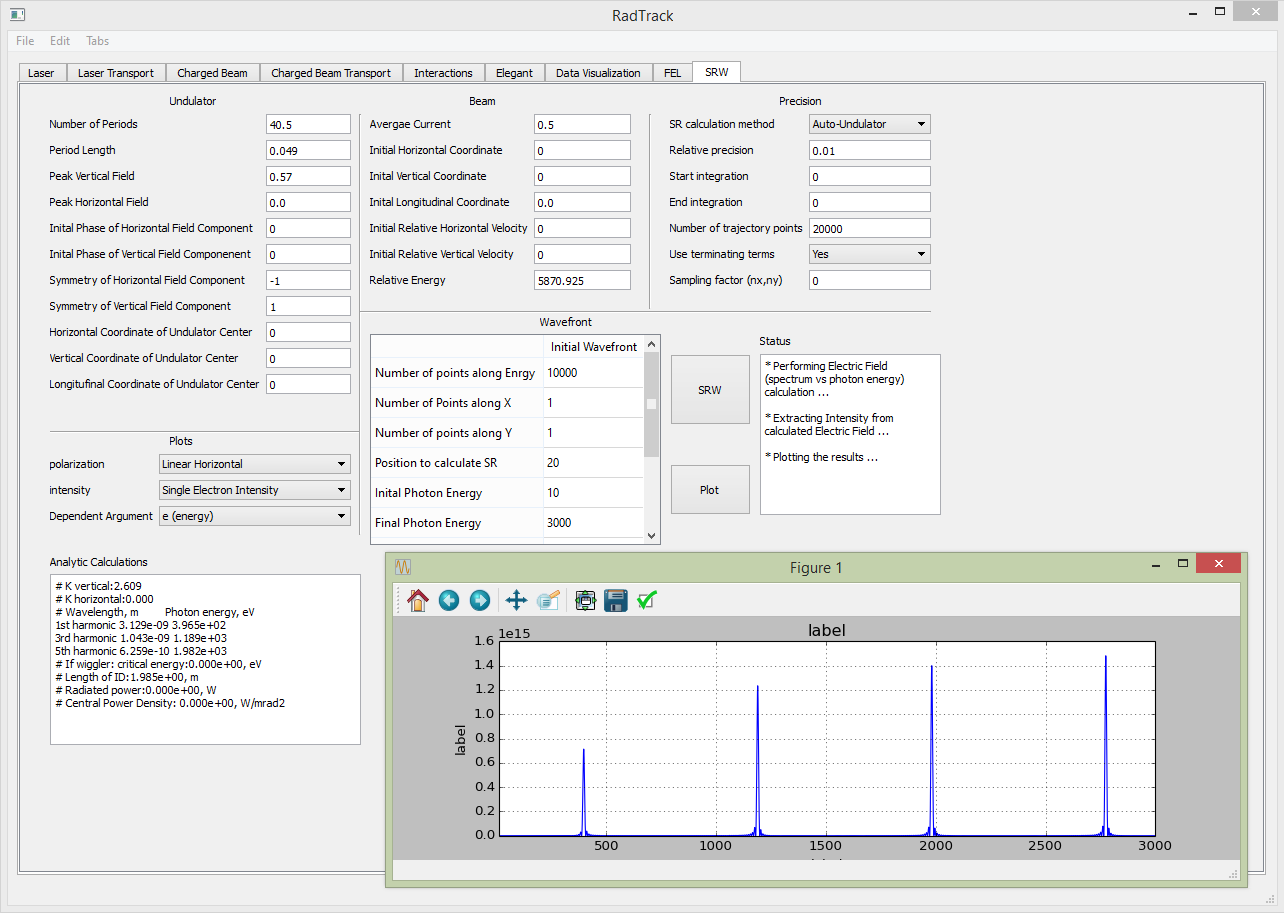
**Figure 28**: Screen shot of the FEL tab, showing an example calculation.

After defining beam and undulator parameters, the user can review the resulting auxiliary problem parameters. These auxiliary parameters provide insight into the details of the problem under study. At the same time, estimated FEL performance is calculated and listed in the ‘FEL Performance’ section of the tab (see right half of Figure 27).

The FEL tab also aids in automatic optimization of a specified FEL property. The user may select one problem parameter to vary, and may assign another parameter with a target value. RadTrack then solves for and returns the independent parameter value that achieves the specified target value. See the bottom-center region of Figure 27.

The plotting window is used to show color contours of how one specified FEL output varies as a function of two specified FEL input parameters. The FEL tab provides very useful estimates and will complement detailed FEL simulation codes, such as GENESIS [[[23]](#endnote-24),[[24]](#endnote-25),39], MEDUSA [[[25]](#endnote-26),[[26]](#endnote-27)], Ginger [[[27]](#endnote-28)] or TDA3D [[[28]](#endnote-29),40].

# The Synchrotron Radiation Workshop (SRW) Tab

This tab provides access to the Synchrotron Radiation Workshop [6,[[29]](#endnote-30),[[30]](#endnote-31)]. There are a number of codes that can model synchrotron radiation and associated optics, but SRW is distinguished by being open source, [[[31]](#endnote-32)] based on the principle of physical optics (as opposed to geometric optics), capable of accurate modeling of fully- and partially-coherent synchrotron radiation emission and propagation through different types of optics, and actively used for development of X-ray and infrared beamlines at light sources. The SRW kernel is written in ANSI C++. Software bindings are available for IGOR Pro, [[[32]](#endnote-33)] a commercial application available for Windows and Mac OSX. Python bindings are available for all platforms, including support for both 32-bit and 64-bit architectures, for Python versions 2.7 and 3.x. Parallel computation for CPU-intensive partially-coherent radiation propagation simulations is implemented from the Python interface, using MPI via the mpi4py [[[33]](#endnote-34)] module. A screen shot of the tab is shown in Figure 28.

**Figure 29**: Screen shot showing the SRW tab.

The underlying Python module includes a beam parameters class with the following data members:

• ex # horizontal emittance [m-rad]

• enx # normalized emittance [m-rad]

• betax # Twiss beta [m/rad]

• alphax # Twiss alpha

• etax # dispersion [m]

• etaxp # dispersion derivative

• ey, eny, betay, alphay, etay, etayp # same as previous 6 lines, for the vertical plane

• Sx, Sxp, Sy, Syp, St, Sdelta # RMS beam sizes, divergences, length, spread

• pAverage, pCentral # average momentum [eV/c]

• el #

• ecx, ecnx, ecy, ecny # corrected emittance, with dispersion subtracted

• betacx, alphacx, betacy, alphacy # and the corresponding Twiss parameters

• Cx, Cxp, Cy, Cyp, Ct, Cdelta # centroid positions in the 6D phase space

• Charge # total bunch charge [C]

• Particles # number of particles

These data correspond to the ‘sddsanalyzebeam’ output of the Elegant/SDDS package [5,[[34]](#endnote-35),[[35]](#endnote-36)]. Therefore, the SRW module in RadTrack uses the same beam definitions and adds more definitions of radiation sampling conditions: waveform parameters, geometrical offsets and initial trajectory conditions. There is also another set of parameters, involving precision and methods of calculation, which are internal to SRW and are never used in other RadTrack tabs.

The results of SRW calculations are exported to files via the same code used by other RadTrack tabs, although a special format called ‘SRW output format’ has been introduced, which contains the problem descriptor, parameters pertaining to the problem and corresponding data matrix with the column labels.

*Problem set-up in the SRW tab*

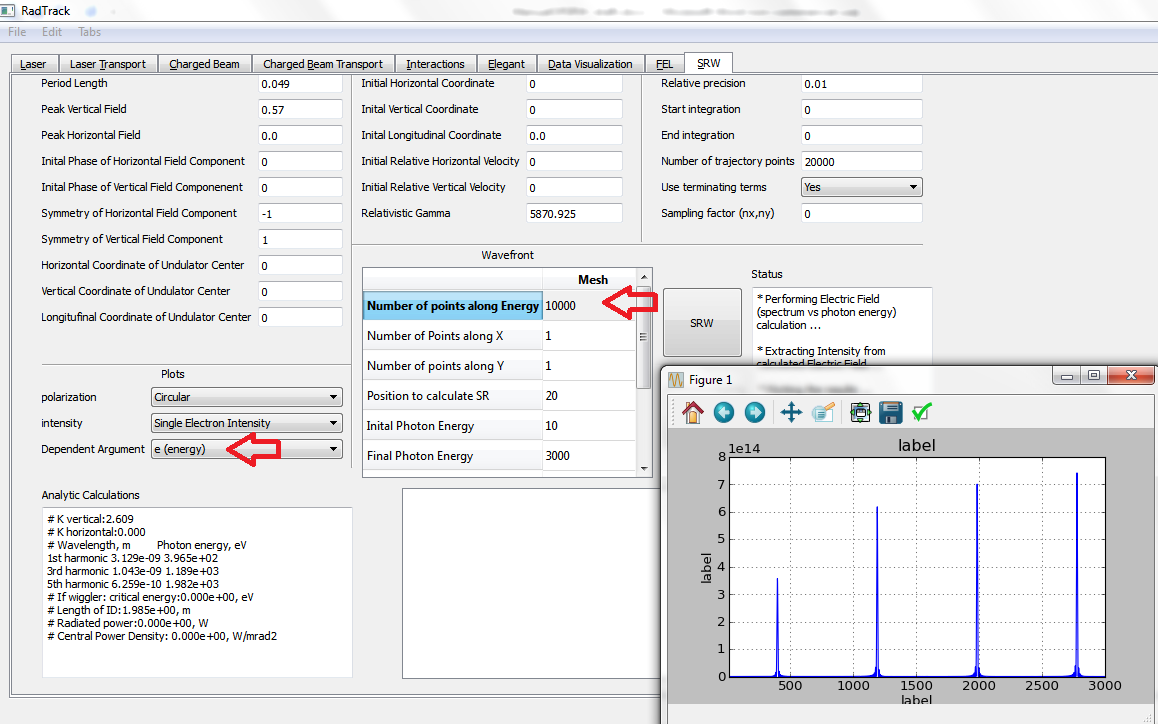
The following parameters must be specified for the SRW tab:

1. Electron beam
   1. energy, current, trajectory, energy offset, Twiss parameters, dispersions, emittances
2. Undulator
   1. period, field strength, length
3. Radiation sampling
   1. distance to image plane, grid dimensions/size, spectral range, number of intervals

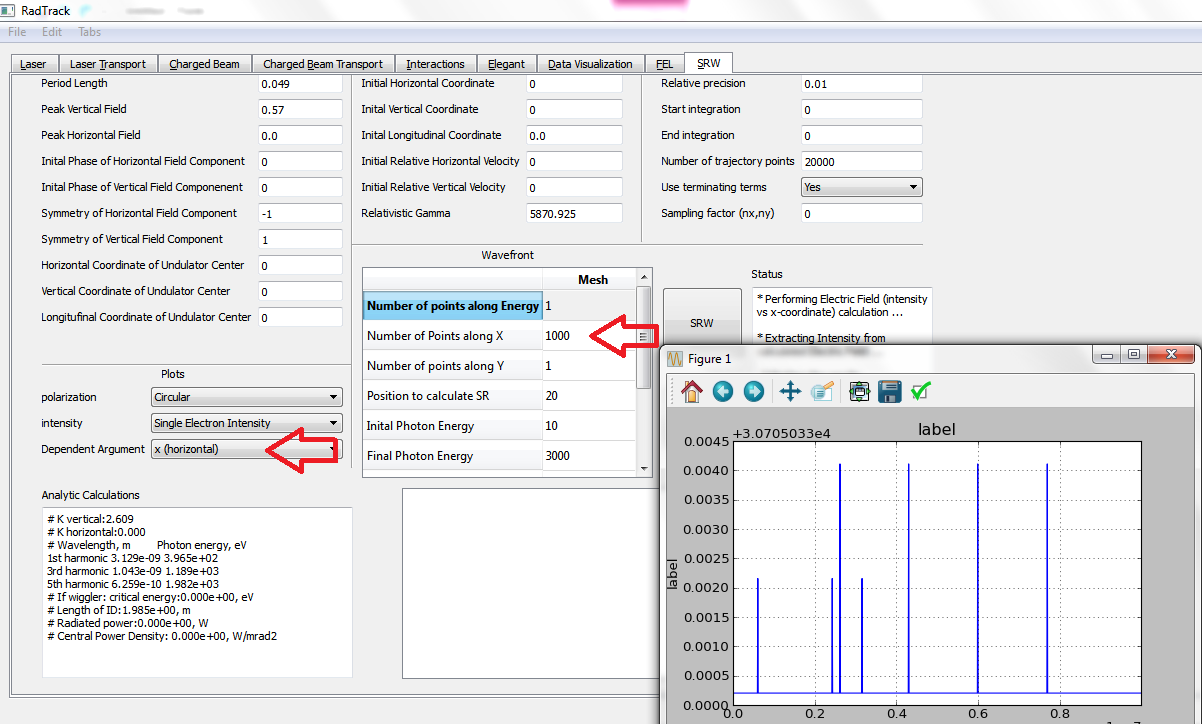
4. Details regarding SRW methods

The definitions are made as illustrated by the layout in Figure 29. Once the problem is defined, the SRW is invoked through its Python API, and the final wavefront is calculated. Next, the user selects properties of the wavefront to be displayed. The main choice is between the spectral and transverse intensity distributions. Various degrees of polarization can be specified as well as flux, intensity, field or phase.

**Figure 30**: Schematic of an SRW simulation of synchrotron radiation from an undulator insert, which then propagates a specified distance in vacuum.

The output is displayed using SRW’s matplotlib features in a separate window. After each simulation, the user can plot various aspects of the calculated wavefront using the ‘Plot’ button. To get a plot of either the transverse profile of the beam or energy spectrum, the user must select more points along the desired dimension, and then select the corresponding dependent argument. For example, to see information regarding the beam energy, the user selects a large number of points along the energy axis and specifies energy as the dependent argument. See Figure 30 for details.

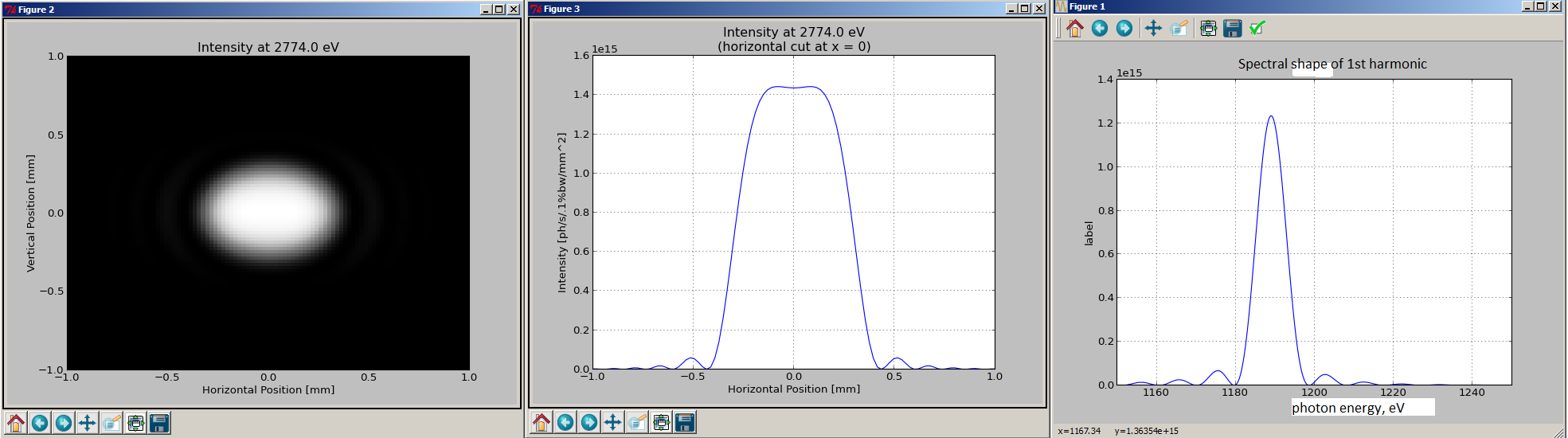
**Figure 31**: Screen shot showing the SRW tab, with red arrows showing how to obtain an energy spectrum plot.

Alternatively, to see information about the transverse beam properties, the user can select more points along the x or y axes and select the corresponding dependent argument. See Figure 31 for details. The SRW “filament” beam model (i.e, zero transverse beam sizes) has been implemented. The more accurate “thick” beam model will be added to the SRW tab in the future.

**Figure 32**: Screen shot showing the SRW tab, with red arrows showing how to plot data as a function of transverse position.

*User interface (UI)*

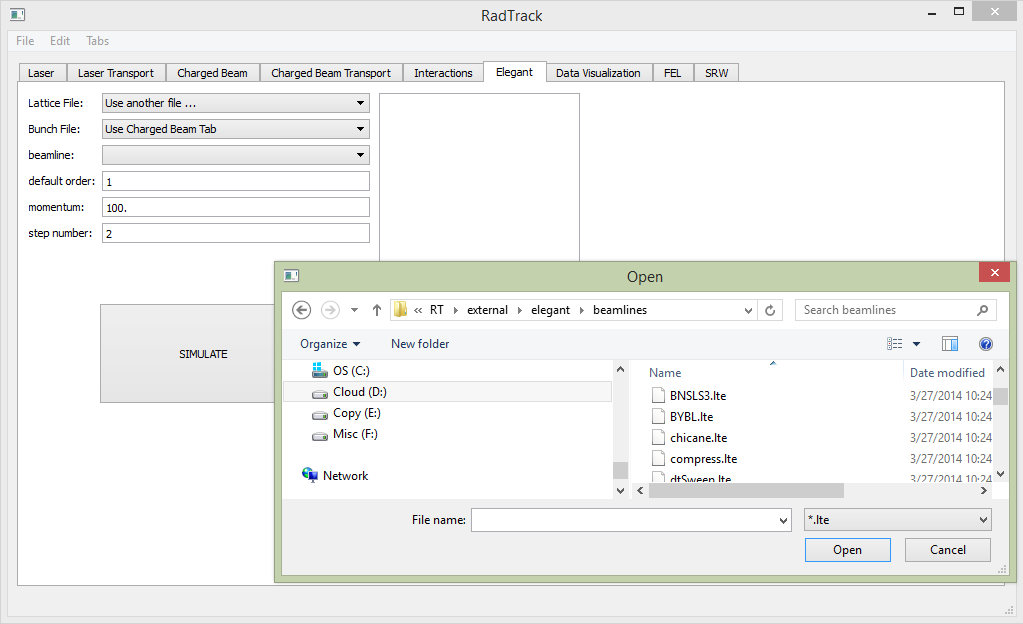
The SRW tab has a UI that facilitates optimization of simple problems like a) undulator radiation, b) sampling of the resulting wavefront through finite apertures, c) calculating heat load and brightness from the radiation fan, and d) studying the impact of source size on the radiation parameters. The plot window includes standard matplotlib features that enable interactive resizing and zooming and saving to file in various formats (png, tiff, etc.). The plots seen in Figure 32 illustrate the 2D transverse intensity distribution, the 1D horizontal projection of the intensity distribution, and the first harmonic of the undulator radiation through a 1 mm x 1 mm aperture located 20 m downstream. These distributions can be saved into SRW style text files for external post-processing.

**Figure 33**: Plots generated by the SRW tab for undulator radiation at an observation point 20 m downstream, after passing through a 1 mm x 1 mm aperture, showing: the 2D transverse intensity distribution (left), the 1D horizontal projection of the intensity distribution (center), and the first harmonic (right).

# The Elegant Tab

This tab provides access to the Elegant code [5,34,35]. Elegant (ELEctron Generation ANd Tracking) is the principle accelerator simulation code used at the Advanced Photon Source (APS) for circular and one-pass machines. Capabilities include 6-D tracking using matrices up to third order, canonical integration, and numerical integration. Standard beamline elements are supported, as well as coherent synchrotron radiation, wakefields, rf elements, kickers, apertures, scattering, and more. In addition to tracking with and without errors, elegant performs optimization of tracked properties, as well as computation and optimization of Twiss parameters, radiation integrals, matrices, and floor coordinates. Orbit/trajectory, tune, and chromaticity correction are supported. elegant is fully compliant with the Self Describing Data Sets (SDDS) file protocol, and hence uses the SDDS Toolkit for pre- and post-processing. This permits users to prepare scripts to run the code in a flexible and automated fashion. It is particularly well suited to multistage simulation and concurrent simulation on many workstations. A screen shot is provided in Figure 33.

Outside of RadTrack, an Elegant simulation is driven by a ‘command’ file, which consists of namelist-like commands. A sequence of commands is required to set up and execute a simulation. Generally, a there are a number of setup commands, followed by a single action command. In addition to the command file, the user must supply at least one separate ‘lattice’ file, similar to what is used by the MAD (Methodical Accelerator Design) code [[[36]](#endnote-37)]. Complex beamlines may be broken into several simple lattice files, then combined via file inclusion.

**Figure 34**: Screen shot showing the Elegant tab.

When running Elegant from within RadTrack, the ‘command’ file is no longer necessary. Setting up an Elegant simulation requires the following inputs:

1. Description of the beamline or accelerator lattice:
   1. standard ‘lattice’ file, with expected file extension ‘lte’
   2. get it internally from the ‘beamline lattice designer’ tab, described above
2. Description of the particle bunch:
   1. an SDDS binary file, containing all macro-particles, with file extension ‘sdds’
   2. get it internally from the ‘charged particle beam’ tab, described above
   3. (not yet implemented) specify Elegant particle distribution generator
3. Simulation control options
   1. use GUI to specify beam momentum & order of transfer matrices
   2. (not yet implemented) specify an Elegant ‘command’ file

To specify an Elegant input file, the user selects ‘use another file’ from the corresponding menu (see Figure 33), then browses the local file system for the necessary file. Users may also generate Elegant input files from other RadTrack tabs (see above). Previously selected files will appear as options in the drop down menu as well. After selecting an Elegant lattice or \*.lte file, the ‘beamlines’ menu will be populated with all ‘beamlines’ in that file, which users can select by name.

When the ‘simulate’ button is selected, a pop-up dialog allows the user to specify a base name for all of the six output files:

1. description of the beam centroid (first moments of the distribution); file extension .cen

2. description of all second moments of the distribution; file extension .sig

3. listing of all magnets in the beamline; file extension .mag

4. ??????; file extension .fin

5. the beam twiss parameters throughout the accelerator lattice; file extension .twi

6. description of the final 6D macro-particle phase space; file extension .out

# Additional capabilities

RadTrack has additional capabilities, which are not yet implemented as a tab in the GUI, but are available in command-line mode. Two significant capabilities are described in detail below. Integration within the tabbed GUI will be implemented in the near future, based on user demand.

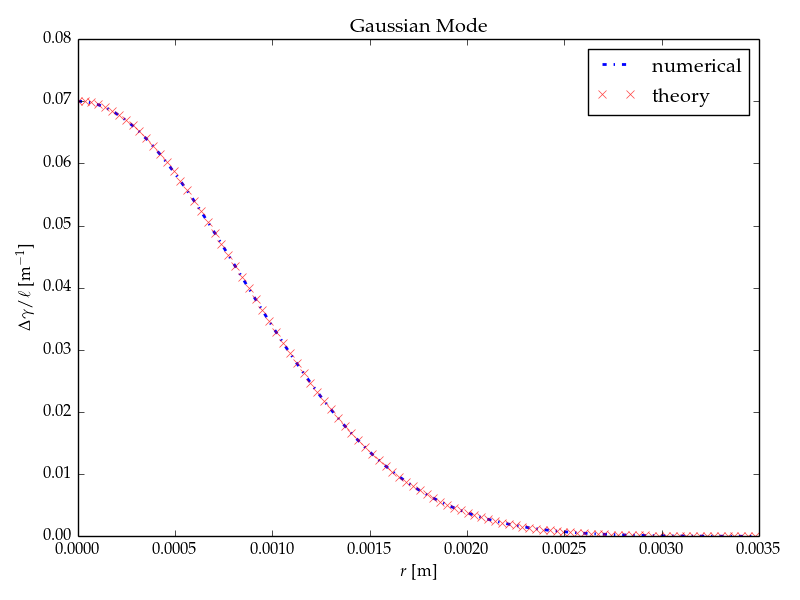
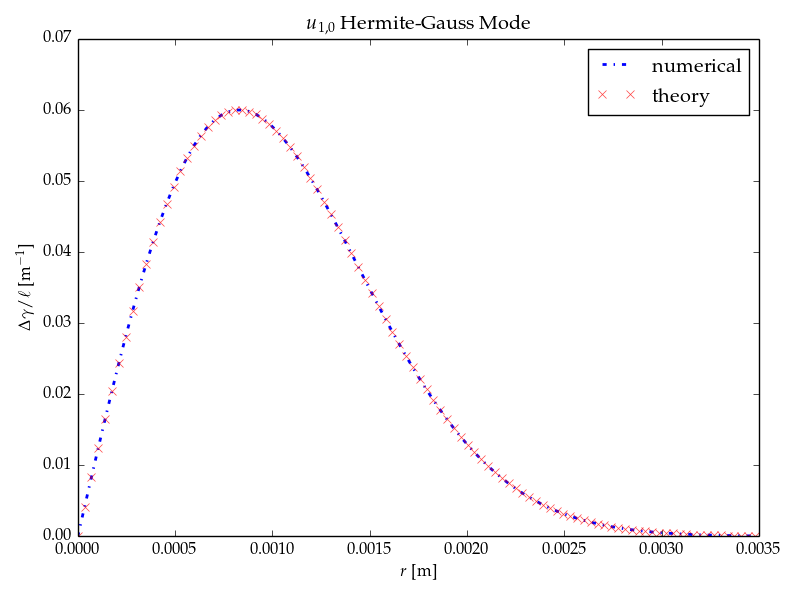
# The Laser Heater Module

The microbunching instability generated by coherent radiation effects in free-electron lasers can be suppressed by increasing the local energy spread of the beam [[[37]](#endnote-38)]. Pioneered at LCLS, the laser-heater is a laser-electron interaction in an undulator that increases the beam energy spread in a controllable way to reduce the impact from collective effects, via a mechanism analogous to Landau damping.

We implemented a pure Python laser heater module in RadTrack, which works with a simple undulator field representation, combined with the full Gauss-Hermite mode expansion of Sec. B.1 above for treating arbitrary laser pulses. The electrons are represented by the Python classes used in developing the charged beam tab, described above in Sec. B.2. The symplectic 2nd-order particle advance of Vay [[[38]](#endnote-39)] has been implemented to advance charged particles through arbitrary fields.

The laser-heater interaction involves resonant FEL electron dynamics in the ponderomotive potential created via beating of the undulator and laser fields. However, there is no FEL instability and no lasing. Hence, the ponderomotive amplitude is fixed in time, assuming the laser pulse is long compared to the laser-electron interaction time, and the laser profiles are approximately constant during this time. The local amplitude of the ponderomotive potential does vary significantly; however, depending on the local amplitude of the laser envelope.

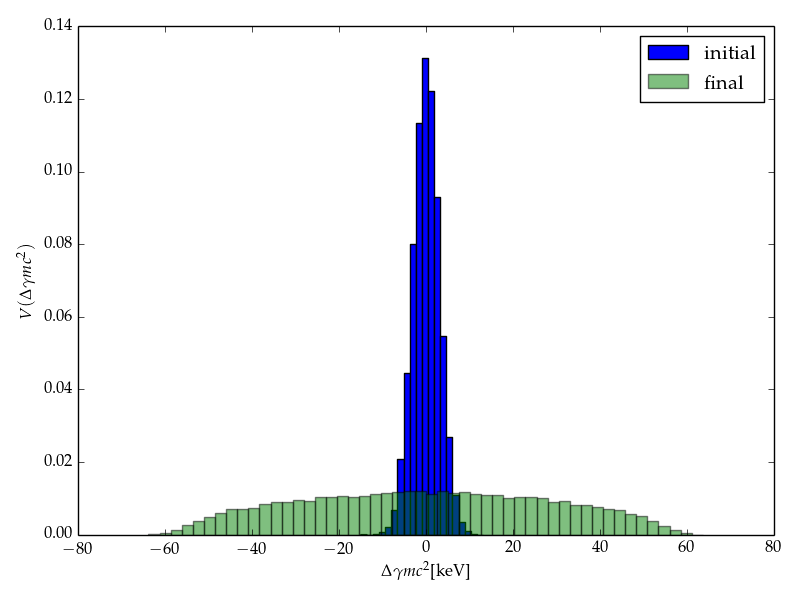
It’s not practical to push electrons through the full, combined, time-explicit laser and undulator fields, so we implemented the well-known (but not really standardized) FEL averaged equations [[[39]](#endnote-40),[[40]](#endnote-41),[[41]](#endnote-42)]. The full implementation was benchmarked against theory – i.e. Eq. (8) of Ref. [37] – which in turn has been used to benchmark the laser-heater algorithm in Elegant [[[42]](#endnote-43)]. Figure 34 shows the change in the relativistic  factor for a single electron passing through a laser-heater system, for the special case of motion parallel to the system axis. The Gaussian shape traces the transverse profile of the laser pulse. Numerical results are obtained from RadTrack, while theory is from Eq. (8) of Ref. [37]. Very close agreement is observed.

**Figure 35** Change in the relativistic  factor for a single electron passing through a laser-heater system, for the special case of motion parallel to the system axis. The Gaussian shape traces the transverse profile of the laser pulse. Numerical results are obtained from RadTrack, while theory is from Eq. (8) of Ref. [37].

**Figure 36:** Change in the relativistic  factor for a single electron passing through a laser-heater system, for the special case of motion parallel to the system axis. The non-Gaussian shape traces the transverse profile of the laser pulse, which here is a (1,0) Gauss-Hermite mode. Numerical results are obtained from RadTrack, while theory is from Eq. (8) of Ref. [37].

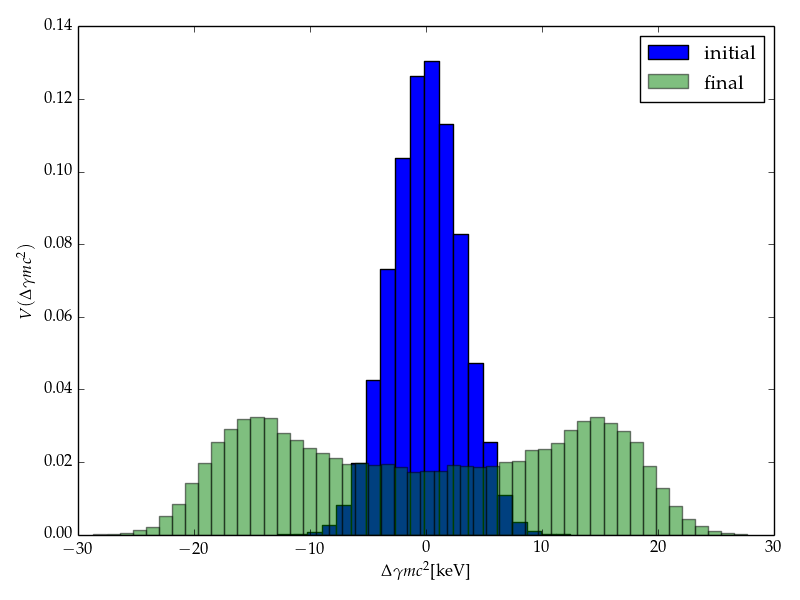
The RadTrack laser-heater module is more general than the Elegant algorithm, which assumes a simple Gaussian laser mode that follows the diffractive dynamics of the paraxial approximation. In contrast, RadTrack can use a Gauss-Hermite expansion to represent arbitrary transverse laser profiles. Figure 35 shows change in the relativistic  factor for a single electron passing through a laser-heater system, for the special case of motion parallel to the system axis. The non-Gaussian shape traces the transverse profile of the laser pulse, which here is a (1,0) Gauss-Hermite mode. Numerical results are obtained from RadTrack, while theory is from Eq. (8) of Ref. [37]. Very close agreement is observed.

The figures above show correct single-particle dynamics for the RadTrack laser heater module. Next, we benchmark results for a full electron bunch. We use parameters from the SLAC Linac Coherent Light Source (LCLS) [[[43]](#endnote-44)], which are taken from Ref. [37], and we compare with Figure 8 from that same paper. The RadTrack results are shown in Figure 36, with the initial electron energy distribution in blue, and the final distribution (after the laser heater) shown in green.

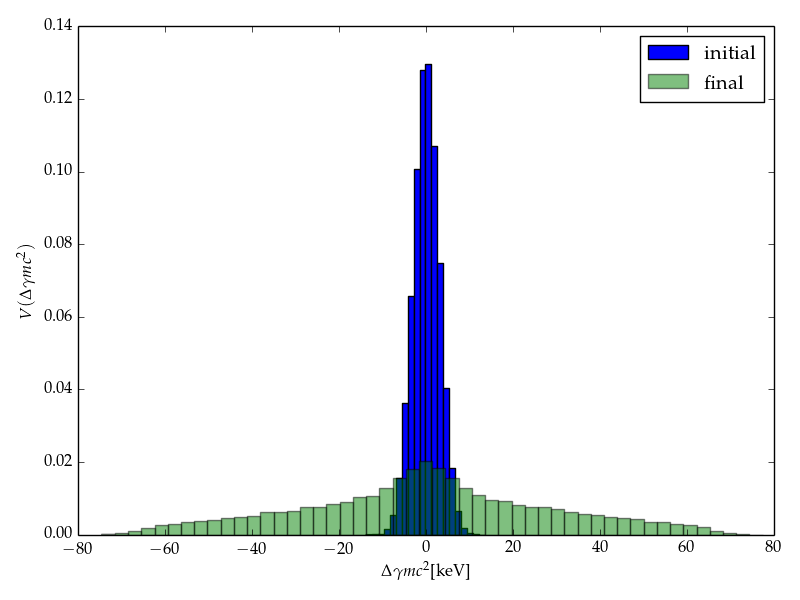
****The blue histogram is consistent with the initial conditions specified by Huang *et al*., and the green histogram shows reasonably good agreement with the red dashed curve of the referenced Figure 8. Huang *et al*. calculate an increase from 3 keV to 40 keV RMS energy spread in the laser heater. The RadTrack simulation shows an RMS energy spread of about 29 keV; however, in this simulation, the Rayleigh length is such that the laser spot size changes significantly along the length of the undulator, so it is to be expected that the simulated energy spread will be lower than the idealized theoretical calculation. Particles that spend time along the edges of the beam are not as strongly affected, because they sample lower (sometimes close to zero) laser electric fields in the middle of the undulator, where the laser pulse is focused to a waist.

**Figure 37:** Results from the RadTrack laser-heater module, with the initial electron energy distribution in blue and the final distribution shown in green. The blue histogram is consistent with the initial conditions specified by Huang et al., and the green histogram shows reasonably good agreement with the red dashed curve of their Figure 8. Here, the RMS energy spread increases from 3 keV to 40 keV.

In Figure 37, the simulation of Figure 36 is repeated but with an order of magnitude longer Rayleigh length, yielding the bimodal distribution observed in Figure 8 of Ref. [37] under similar conditions. Here, the laser power was not increased to maintain a constant peak electric field for the increased waist size. Hence, the E-fields are smaller by roughly the square root of the ratios of the two Rayleigh lengths – in this case a factor of three, which explains why the limits of our green histogram (~20 keV) are about 3x smaller than the limits of the blue curve in Huang *et al*. (~60 keV).

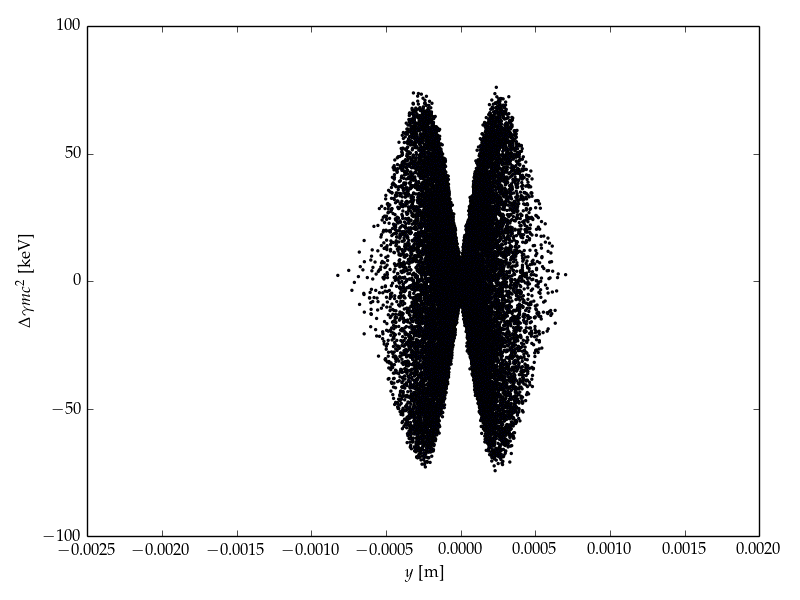
**Figure 38**: The simulation of Figure 36 is repeated but with an order of magnitude longer Rayleigh length, yielding the bimodal distribution observed in Figure 8 of Ref. [37] under similar conditions.

In Figure 38, the simulation of Figure 36 is repeated but with a (1,0) Gauss-Hermite mode, for which the electric field goes to zero on the system axis. The resulting energy distribution, shown by the green histogram, is different in the details but not qualitatively changed. This result is reasonable, because the integrated heating of the electron distribution depends on the average value of the laser electric field across the electron beam cross-section. This average value is comparable for the simulation parameters used to generate these two figures.

**Figure 39:** The simulation of Figure 36 is repeated with a (1,0) Gauss-Hermite mode, for which the on-axis E-field goes to zero on the system axis. The resulting energy distribution, shown by the green histogram, differs in the details, but is not qualitatively changed.

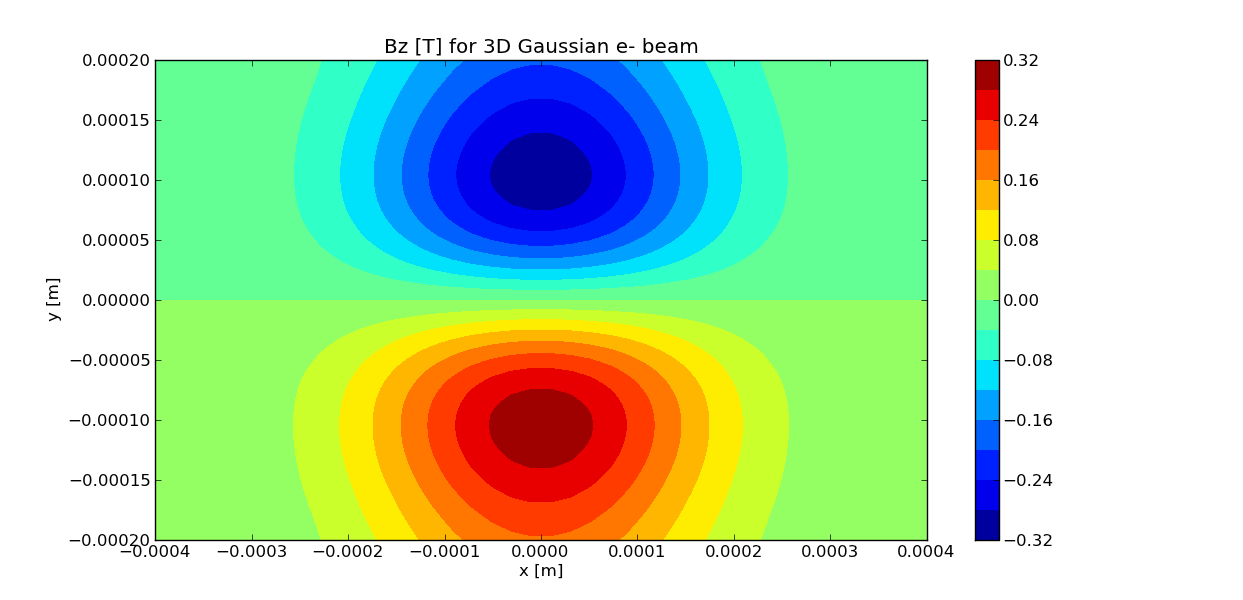
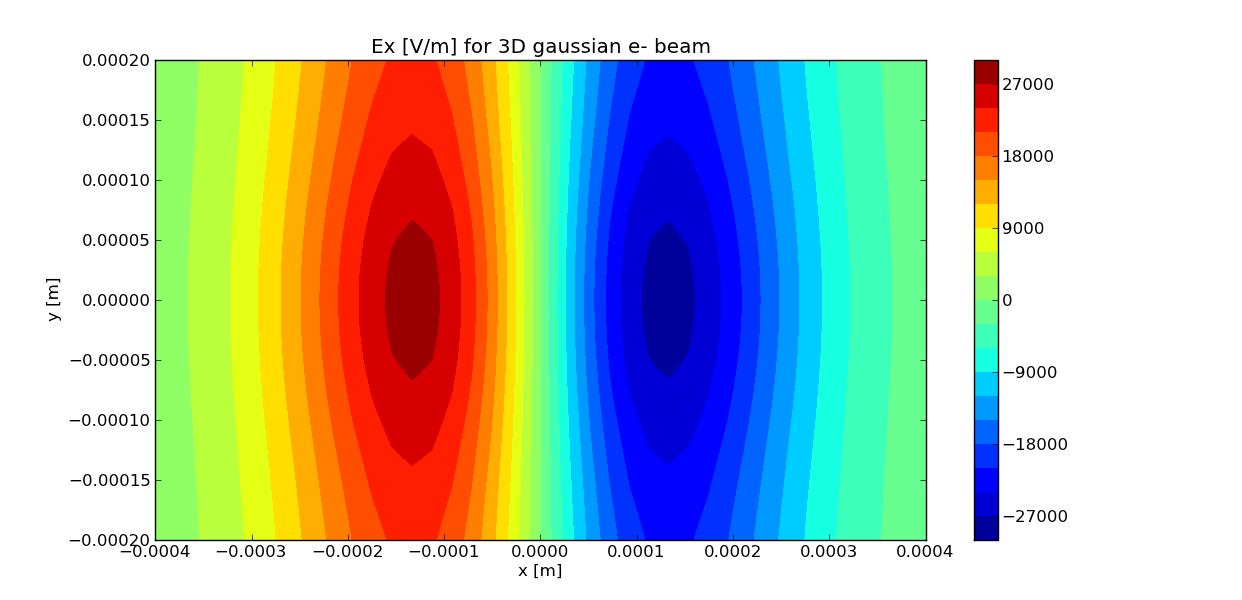
****However, striking differences in the heating generated by the Gauss-Hermite mode can be seen in 2D scatter plots of energy vs horizontal and vertical position. Figure 39 shows how the energy spread depends on horizontal position, with a simple Gaussian shape. In stark contrast, Figure 40 reveals the shape of the 1st-order Gauss-Hermite mode in how the energy spread varies with vertical position.

**Figure 40:** Scatter plot of heated electrons from the simulation of Figure 38, showing the expected Gaussian variation of thermal spread with horizontal distance from the axis.

**Figure 41:** The non-Gaussian variation of thermal spread with vertical distance from the axis, seen in this 2D scatter plot of heated electrons from the simulation of Figure 38, reveals the shape of the 1st-order Gauss-Hermite mode in the simulated laser pulse.

# The Relativistic Beam Wakefield Module

RadTrack includes an algorithm to calculate the quasistatic electric and magnetic wakefields of 3D Gaussian particle beams in free space. Taking the lab frame properties of a beam (energy, charge, rms dimensions), the 3D distribution is Lorentz-transformed into the beam frame, where the relative particle motion is assumed to be non-relativistic (correct for high-quality beams) and the Poisson equation is solved via 1D quadrature. These E-fields are Lorentz-transformed back to the lab frame, to obtain E and B. The algorithm assumes the beam is evolving slowly (correct for a high-quality beam in a conventional accelerator lattice), which we refer to as a quasistatic approximation. The algorithm also assumes a Gaussian distribution in all three spatial dimensions. Figure 41 provides an example.



**Figure 42:** Ex (longitudinal, left) and Bz (perpendicular, right) are shown for a 50 MeV, 0.1 nC electron beam, with 1 ps FWHM length and 50 m rms width.

# Technical References

1. . The Python programming language; <https://www.python.org/> [↑](#endnote-ref-2)
2. . Website for Qt, a cross-platform application and UI framework; <http://qt-project.org/> [↑](#endnote-ref-3)
3. . The PySide interface between Python and Qt; <http://qt-project.org/wiki/PySide> [↑](#endnote-ref-4)
4. . Matplotlib 1.3.1 documentation; <http://matplotlib.org> [↑](#endnote-ref-5)
5. . M. Borland, “elegant: A flexible SDDS-compliant code for accelerator simulation,” APS Technical Report, LS-287 (2000); <http://www.ipd.anl.gov/anlpubs/2000/08/36940.pdf> [↑](#endnote-ref-6)
6. . O. Chubar and P. Elleaume, “Accurate and Efficient Computation of Synchrotron Radiation in the Near Field Region,” Proc. European Part. Accel. Conf., p.1177 (1998). [↑](#endnote-ref-7)
7. . D. Schiller et al., “QUINDI – A code to simulate coherent emission from bending systems”, Proc. Part. Accel. Conf., p. 3612 (2007). [↑](#endnote-ref-8)
8. . QUINDI website: <http://pbpl.physics.ucla.edu/Computing/Code_Development/QUINDI/> [↑](#endnote-ref-9)
9. . The Qt creator; <http://qt-project.org/wiki/QtCreatorWhitepaper> [↑](#endnote-ref-10)
10. . J. Alda, “Laser and Gaussian Beam Propagation and Transformation,” Encyclopaedia of Optical Engineering (Marcel Dekker, New York, 2002). [↑](#endnote-ref-11)
11. . F. Pampaloni and J. Enderlein, “Gaussian, Hermite-Gaussian, and Laguerre-Gaussian beams: A primer” (2004); <http://arxiv.org/abs/physics/0410021> [↑](#endnote-ref-12)
12. . A.E. Siegman, Lasers (Univ. Science Books, 1986), Chapter 16. [↑](#endnote-ref-13)
13. . Wikipedia page on paraxial laser modes; <https://en.wikipedia.org/wiki/Gaussian_beam> [↑](#endnote-ref-14)
14. . SciPy website; <http://www.scipy.org/> [↑](#endnote-ref-15)
15. . NumPy website; <http://www.numpy.org/> [↑](#endnote-ref-16)
16. . Handbook of Accelerator Physics and Engineering, 2nd Edition, Eds. A.W. Chao, K.H. Mess, M Tigner and F. Zimmermann (World Scientific, 2013), Sections 2.1 and 2.2. [↑](#endnote-ref-17)
17. . R. Soliday, M. Borland, L. Emery and H. Shang, “New features in the SDDS toolkit,” Proc. Part. Accel.Conf., 3473 (2003); <http://epaper.kek.jp/p03/PAPERS/FPAG005.PDF> [↑](#endnote-ref-18)
18. . H. Shang, Y. Wang, R. Soliday, M. Borland and L. Emery, “Parallel SDDS: A Scientific High-Performance I/O Interface,” Proc. Int. Comp. Accel. Phys. Conf., THPSC050 (2009);

    <http://accelconf.web.cern.ch/Accelconf/ICAP2009/papers/thpsc050.pdf> [↑](#endnote-ref-19)
19. . W. Herr and F. Schmidt, “Using SDDS Data Sets with MAD-X,” CERN AB-Note-2006-043 (2006); <http://cds.cern.ch/record/996174/files/ab-note-2006-043.pdf> [↑](#endnote-ref-20)
20. . The SDDS download page;

    <http://www.aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/software.shtml#SDDSSource> [↑](#endnote-ref-21)
21. . D.L. Bruhwiler, O. Chubar, R. Nagler, J. Krzywinski and A. Boehnlein, “An open software framework for advancement of X-ray optics simulation and modeling,” Proc. SPIE Workshop: Advances in Computational Methods for X-Ray Optics (San Diego, August 2014). [↑](#endnote-ref-22)
22. . M. Xie, “Design Optimization for an X-Ray Free Electron Laser Driven by SLAC Linac,” Proc. Part. Accel. Conf., 183 (1996); <http://accelconf.web.cern.ch/AccelConf/p95/ARTICLES/TPG/TPG10.PDF> [↑](#endnote-ref-23)
23. . S. Reiche, “GENESIS 1.3: a fully 3D time-dependent FEL simulation code,” Nucl. Instrum. and Methods A **429**, 243 (1999). [↑](#endnote-ref-24)
24. . GENESIS website; <http://genesis.web.psi.ch> [↑](#endnote-ref-25)
25. . H.P. Freund, “Time-dependent simulation of free-electron laser amplifiers and oscillators,” Phys. Rev. Special Topics – Accel. & Beams **8**, 110701 (2005). [↑](#endnote-ref-26)
26. . H.P. Freund and T.M. Antonsen, Jr., Principles of Free Electron Lasers (Chapman & Hall, 1986). [↑](#endnote-ref-27)
27. . W.M. Fawley, “A User Manual for GINGER and its Post-Processor XPLOTGIN,” Report LBNL-49626 (2001); <http://escholarship.org/uc/item/9595m1wv> [↑](#endnote-ref-28)
28. . B. Faatz, W. Fawley, P. Pierini, S. Reiche, G. Travish, D. Whittum and J. Wurtele, “TDA3D: Updates and improvements to the widely used three-dimensional free electron laser simulation,” Nucl. Instr. & Meth. in Phys. Research A **393**, p. 277 (1997). [↑](#endnote-ref-29)
29. . The SRW page at ESRF; <http://www.esrf.eu/Accelerators/Groups/InsertionDevices/Software/SRW> [↑](#endnote-ref-30)
30. . O. Chubar, A. Fluerasu, L. Berman, K. Kaznatcheev and L. Wiegart, “Wavefront propagation simulations for beamlines and experiments with ‘Synchrotron Radiation Workshop’,” J. Phys.: Conf. Ser. **425**, 162001 (2013); <http://iopscience.iop.org/1742-6596/425/16/162001> [↑](#endnote-ref-31)
31. . The SRW repository on Github; <https://github.com/ochubar/SRW> [↑](#endnote-ref-32)
32. . The IGOR Pro website; <http://www.wavemetrics.com/products/igorpro/igorpro.htm> [↑](#endnote-ref-33)
33. . The MPI for Python module; <http://mpi4py.scipy.org> [↑](#endnote-ref-34)
34. . Argonne National Lab download site for Elegant, SDDS and other codes;

    [www.aps.anl.gov/Accelerator\_Systems\_Division/Accelerator\_Operations\_Physics/software.shtml](http://www.aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/software.shtml) [↑](#endnote-ref-35)
35. . M. Borland, L. Emery, H. Shang and R. Soliday, “SDDS-Based Software Tools for Accelerator Design,” Proc. 2003 Part. Accel. Conf., p. 3461 (2003);

    <http://accelconf.web.cern.ch/accelconf/p03/PAPERS/FPAG001.pdf> [↑](#endnote-ref-36)
36. . MAD-X reference [↑](#endnote-ref-37)
37. . Z. Huang, M. Borland, P. Emma, J.Wu, C. Limborg, G. Stupakov and J.Welch, “Suppression of microbunching instability in the linac coherent light source,” Phys. Rev. Special Topics – Accel. & Beams **7**, 074401 (2004). [↑](#endnote-ref-38)
38. . J.-L. Vay, “Simulation of beams or plasmas crossing at relativistic velocity,” Phys. Plasmas **15**, 056701 (2008). [↑](#endnote-ref-39)
39. . S. Reiche, “Numerical Studies for a Single Pass High Gain Free-Electron Laser,” U. Hamburg PhD Dissertation, DESY-THESIS-2000-012 (2000).

    <http://www-library.desy.de/cgi-bin/showprep.pl?desy-thesis-00-012> [↑](#endnote-ref-40)
40. . T.M. Tran and J.S. Wurtele, “TDA - A three-dimensional axisymmetric code for free-electron-laser (FEL) simulation,” Comp. Phys. Comm. **54**, p. 263 (1989). [↑](#endnote-ref-41)
41. . E. Saldin, E.A. Schneidmiller & M.V. Yurkov, The Physics of Free Electron Lasers (Springer, 2000). [↑](#endnote-ref-42)
42. . Elegant user manual (for the laser heater feature, see Sec. 10.42 LSRMDLTR, p. 229);

    <http://aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/manuals/elegant_latest/elegant.pdf> [↑](#endnote-ref-43)
43. . The LCLS website; <https://portal.slac.stanford.edu/sites/lcls_public/Pages/Default.aspx> [↑](#endnote-ref-44)