elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation *

M. Borland, ANL, Argonne, IL 60439, USA

Abstract

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 [1, 2] (SDDS) file protocol, and hence uses the SDDS Toolkit for pre- and postprocessing. 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. Several examples of complex projects performed with elegant are given, including top-up safety analysis of the APS and design of the APS bunch compressor.

1 INTRODUCTION

There is no shortage of accelerator codes in the world today, so it seems that anyone presenting a new accelerator code needs to justify its creation. The code should provide new or better algorithms, a new or better interface, improved throughput, or some other clear benefit.

elegant (ELEctron Generation ANd Tracking) was created in response to specific needs of the author that were unmet by codes that existed at the time. Since then, it has grown through meeting the challenges of designing and commissioning a number of accelerators at the Advanced Photon Source (APS). A consistent emphasis in developing elegant was to allow easily performed simulated experiments that mimic those one might perform on a real accelerator. For example, elegant can vary accelerator parameters in nested loops and track to find the variation of beam properties.

In addition, elegant is fully compliant with the Self-Describing Data Sets (SDDS) data file protocol, which gives users access to a suite of about 70 generic data processing and display tools. These tools can be used together with scripting languages like Tcl to compose customized postprocessing commands. This permits, for example, users of elegant to perform an arbitrary number of similar simulations without additional effort in postprocessing. In addition, elegant itself does not have or need a postprocessor, which makes code maintenance and development easier.

Contrary to recent trends, elegant does not have a graphical user interface. I firmly believe that such interfaces to simulation codes are unproductive in a research environment. A scripting interface is much more powerful and more suited to the research physicist's needs. While elegant does not have a script language itself, coupling elegant with SDDS and common external script languages gives the same benefits.

The remainder of this paper is divided into five parts: 1. The architecture and flow of elegant as seen by the user. 2. A review of the general capabilities of elegant, including the types of analyses performed by elegant. 3. A review of accelerator elements supported by elegant. 4. A review of output files produced by elegant and how to obtain them. 5. The SDDS Toolkit and how it is used with elegant. 6. Examples. This paper does not cover details of commands and element definitions; for this, the user should consult the manual [3].

2 ARCHITECTURE AND FLOW

An elegant run is driven by a command file. This file consists of namelist-like commands and optional comments. A sequence of commands is generally required to set up and execute a calculation, such as tracking a beam. Generally, a sequence consists of a number of "setup" commands followed by a single "action" command that executes the calculations specified in the setup commands. Several such sequences may be given in a single input file. Sequences do not communicate with each other, except by using external files to store and retrieve data. For brevity in what follows, I'll refer to a command sequence as a "run," even though an actual run may contain many such sequences.

In addition to the command file, the user must supply at least one separate lattice file. The lattice file is similar in structure to those used by the program MAD (Methodical Accelerator Design) [4], with some differences due to variations in capabilities and the types of elements available. Complex lattice files may be broken into simple lattice files and then combined using a file inclusion facility.

Output from elegant takes two forms. The first is text output that is intended to inform the user of the progress of the simulation. The second is one or more SDDS files that the user requests elegant to create. Requests for the creation of SDDS output files are made through the namelist commands in the command file, and in some cases through definitions of elements in a lattice file. Examples of SDDS

* Work supported by the U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

output files include final and intermediate beam coordinates from tracking, parameters of elements, and Twiss parameters.

Input to elegant, other than the command and lattice files, takes the form of SDDS files. These are frequently SDDS files written by another elegant run, but may have any source. Examples are input of beam coordinates for tracking, element perturbation data, and starting Twiss parameters.

elegant has no embedded postprocessing or graphics support. All such tasks are delegated to the SDDS Toolkit, which provides a much more powerful system than could reasonably be embedded in a physics code. In addition, the SDDS Toolkit serves as a postprocessor for a number of other codes, removing the need to write and maintain dedicated postprocessors. A complex simulation typically consists of an elegant command file, a lattice file, and a script file containing SDDS commands to postprocess and display the results of the run.

3 GENERAL CAPABILITIES

3.1 Alteration of Accelerator Elements

Alteration of accelerator elements refers to changing the parameters of an accelerator component after it is defined by the lattice file. elegant has the ability to simulate multiple instances of the same system within a single run. These instances may differ through addition of random errors to element parameters; user-specified, regular variation of parameters; variation of parameters through loading from an external file; variation of the initial beam for tracking; or various combinations of these.

This capability introduces two possibilities for some of the physics commands: the possibility of immediate output of computations for the unperturbed or unaltered system, and the possibility of output for each instance of the perturbed or altered system. For example, one might simulate a storage ring with focusing errors. One could request Twiss parameter computations for the ideal ring, or for the ring in each perturbed case.

1. Altering parameters directly. The alter_elements command permits altering a parameter of one or more elements to have a new value. For example, one could readily compose a command to change the sextupole component of a set of dipoles for a specific run, without having to alter the lattice file.

2. Loading parameters from external files. The load_parameters command allows loading values of parameters of elements from an SDDS file. This may be done once, to alter the starting definition of the lattice, or repeatedly, to load successive instances of the lattice. An example of the former would be to load ideal quadrupole strengths from a matching run into a tracking run. An example of the latter would be to load successive sets of errors; this can be useful if errors need to be computed in a way that elegant does not support internally (e.g., an unusual statistical distribution or complex linkage of different errors).

3. Addition of random errors to parameters of elements. This is accomplished using the error_element command, which supports the uniform and gaussian distributions with user-specified cutoff. The user may "bind" errors for like-named elements together, or allow them to be different. Error values may be saved to an SDDS file for later reloading into the same or another run using load_parameters.

4. Variation of parameters in loops. The vary_element command allows specifying regular variation of parameters of elements. Any number of looping indices may be defined, for multidimensional parameter sweeps. The variation is linear and equispaced by default, but may also be geometric. In addition, a sequence of values may be loaded from an SDDS file.

5. Linking of parameters of elements. The link_elements command allows specifying values of parameters of one element in terms of values of parameters of another element. In MAD, this type of relationship would be declared in the lattice file. elegant permits specifying the position of the "source" element relative to the element to be changed, which is useful when there are multiple elements with the same name. For example, misalignments of many sets of elements on a series of girders could be linked to the first element on each girder.

3.2 Saving Accelerator Element Information

elegant can alter element parameters in various ways, some of which were outlined in the previous subsection. There are two ways to save this information. The save_lattice command writes a new lattice file that can be read by another elegant run. The file contains all of the elements in the original file, but with updated values for parameters.

The save_parameters command writes an SDDS file that can be used with load_parameters to restore the configuration of the lattice at the time the save was made. This is useful primarily in cases where one wants to selectively restore parts of a configuration, or where one wants to cycle through a series of configurations and perform similar calculations for each. Since save_parameters creates an SDDS file, the SDDS Toolkit can be used for manipulation of the data, such as selection of subsets.

Neither of these facilities saves the perturbed state of a lattice. Instead, they save the reference state of the lattice, or the lattice "definition." The lattice definition is changed by operations such as matching, altering elements, and (if desired) loading parameters. It is not changed by random errors or variation of parameter in loops.

3.3 Orbit and Trajectory

elegant has several commands that are related to computation and correction of closed orbits and trajectories. The closed_orbit command computes closed orbits and writes the orbits to an SDDS file. The user can control

the convergence parameters of the algorithm, which is useful when the working point is near an integer. The user may also specify that the orbit length should be fixed to the length of the ideal orbit; this is a more realistic computation than what is done in most codes, where the rf frequency of the storage ring is implicitly assumed to change to compensate for the path-length change.

The `correct` command corrects orbits or trajectories. A number of SDDS output files are provided, giving orbit/trajectory and corrector data and statistics. The user may specify which elements to use as correctors; for example, quadrupole position may be used for steering. As in the `closed_orbit` command, the user may select fixed-length orbit calculations. Noise may be added to beam position monitor readings to give more realistic results for the accuracy of correction.

The correction matrix used for orbit or trajectory correction may be output to an SDDS file using the `correction_matrix_output` command. Both the response and the inverse matrix are available, with and without the fixed-length constraint. (This output is used in the APS controls system for correction of the orbit in the APS ring.)

Finally, `elegant` provides analysis of orbit amplification factors for corrected and uncorrected cases. This allows, for example, determining how large an orbit perturbation will result from offsets of individual quadrupoles in the presence and absence of orbit correction. It also provides the kick strength that is required by each perturbation for each corrector.

3.4 Optics Calculations and Correction

`elegant` provides several types of optics calculations, all available either for the reference or perturbed lattice. These include Twiss parameters, radiation integrals, chromaticity, and transport matrices. These are provided by the `twiss_output` and `matrix_output` commands. Computation of chromaticity and nonlinear matrix data is controlled by the `default_order` control in the `run_setup` command, and also by the `ORDER` parameter on individual elements. It is the user's responsibility to set these parameters correctly to get data for the situation of interest. By default, these calculations are performed to second order.

`elegant` provides for correction of the tune and chromaticity, using the `correct_tunes` and `chromaticity` commands. In both cases, the user provides the names of two families of quadrupoles or sextupoles, where a family is a set of elements with the same name. If more than two families are involved, additional families may be linked to the primary families (or elements) using `link_elements`. By default, neither tune nor chromaticity correction changes the reference lattice, but rather changes the lattice used for a particular simulation step. This is the desired mode when simulating many successive randomized machines with correction. One may cause the reference lattice to be changed using the `change_defined_values` control on these commands.

A different type of optics calculation supported by `elegant` is inference of a first-order transport map from tracking. This calculation, performed by the `analyze_map` command, is primarily useful for debugging new elements.

3.5 Optimization

A series of commands support the optimization feature of `elegant`. These include `optimization_setup`, `optimization_term`, `optimization_variable`, and `optimize`. All but the last of these are setup commands that prepare for optimization.

`optimization_setup` is used to define the general parameters of the optimization, including the method (Simplex is preferred), whether to minimize or maximize, how many evaluations to perform, how frequently to provide output, and, optionally, the optimization penalty function. Unlike most other codes, the user must provide `elegant` with the optimization penalty function. This provides considerable flexibility, but can be more complicated. In order to make it easier to create the penalty function, the `optimization_term` command may be used to specify individual terms in the function; the terms are summed to obtain the total penalty function.

Depending on what computations the user has requested prior to invoking the optimizer, the optimization equation may refer to properties of the tracked beam (e.g., emittance or centroid position); Twiss parameters at interior points or at the end of the beamline; overall lattice properties (e.g., maximum beta function, equilibrium emittance, chromaticity); first- and second-order matrix elements at the end of the beamline; and floor coordinates at the end of the beamline. The ability to refer to so many properties of the lattice and the tracked beam gives `elegant` the capability to perform optimizations that other codes can't. For example, `elegant` has been used to directly optimize the emittance of the APS ring or to optimize the energy spread of a set of macroparticles in a linac with wakefields.

3.6 Tracking

Tracking in `elegant` requires definition of a beam and a tracking method. Definition of the beam is performed with the `bunched_beam` or `sdds_beam` commands. The former permits internal generation of a beam with a given emittance, Twiss parameters, bunch length, and energy spread. Various distributions are supported, such as gaussian, uniform, and shell.

The `sdds_beam` command permits loading particle data from an SDDS file. This SDDS file may be generated by `elegant` itself, using the `output` parameter of the `run_setup` command to save the final coordinates from tracking. It may also be generated by a `WATCH` element, which provides phase-space output at a point in the lattice. Finally, the SDDS file may be generated by another program or script.

For both the bunched_beam and sdds_beam command, elegant can track multiple times with the same bunch, or it can track different bunches in succession. For the sdds_beam command, the latter requires having multiple pages in the input file.

To invoke tracking, the user gives the track command. Generally, this command has no arguments. The method employed for tracking particles then depends on the particular elements used. For example, the user is free to employ a symplectic implementation of a dipole together with a first-order implementation of a quadrupole.

Other tracking controls appear in other commands. For example, the number of instances to track, the order of tracking, and other controls are provided through the run_setup command. However, track provides some options, such as tracking longitudinal coordinates only in a storage ring and tracking with a "linear chromatic" matrix. The latter allows tracking with chromatic effects but no other nonlinearities. These two modes are exceptions to the general rule in elegant, namely, that tracking methods are determined on an element-by-element basis.

A similar control is the concat_order parameter of the run_setup command. It can be used to force concantenation of elements simulated with matrices into a single matrix. In no case is a higher-order element concatenated to a lower order, however. Instead, elegant concatenates into a series of matrices of the request order, interspersed with matrices of higher order and other elements that are not implemented as matrices.

3.7 Miscellaneous

elegant performs other tasks that do not fit well into the above categories. The find_aperture command can search for the aperture of a machine, whether dynamic, physical, or a combination (e.g., horizontal "dynamic" aperture limited by coupling into the vertical in the presence of a small vertical aperture).

Free Electron Laser (FEL) calculations are performed using the sasefel command, which starts from the properties of the tracked beam and uses the parametrization of Ming Xie [6]. The computations may be performed for a user-specified number of beam slices, where each slice contains the same number of particles. The gain length and other FEL properties may be used in the optmizer.

The floor_coordinates command may be used to obtain floor coordinate output to a file and for use with the optimizer. At present, elegant does not compute floor coordinates correctly for beamlines involving vertical bends.

The subprocess command permits executing a command in the native command shell (e.g., UNIX shell). Such "shell commands" typically preceed the main run, providing processing of input data for run setup, or follow the main run, providing processing of output data. Shell commands may also be used anywhere in namelist command, so that, for example, the result of an SDDS Toolkit shell command may be used as a parameter of a namelist command. The syntax for this is to give a sequence like "{command}" instead of the normal value for the namelist entry, where command is any command (SDDS-based or otherwise) that can be executed in your shell.

Finally, the run_setup and run_control commands provide overall control of the simulation. Many of the output files are specified by run_setup, as is the lattice filename, the beamline name, the central momentum, the default tracking order, the matrix concatenation order (if any), and the random number seed. run_control is used to specify the number of configurations or beams to generate and track, the time spacing of successive bunches, the number of passes for circular machines, and the number of indices for variation of parameters.

At this point, the reader may well be confused and will certainly not be able to use elegant after reading the above. The best way to learn to use elegant is by studying examples, a number of which are provided on our software distribution Web page.

4 ACCELERATOR ELEMENTS

elegant supports about 75 different accelerator elements. (I say "about" because some are mere place-holders while others are obsolete.) Parameter lists, data types, and units are listed in the manual. Here, I simply describe the elements in general terms.

The common beam-transport elements are supported using a matrix implementation up to second order. This includes drift spaces, dipoles, quadrupoles, sextupoles, solenoids, and correctors. In addition, alpha magnets are supported up to third order [7]. Matrix concatenation is supported up to third order as well.

For tracking circular machines, it is well known that second-order matrix tracking is not adequate for dynamic aperture and other applications requiring many turns. elegant provides two options to address this problem. First, one may explicitly change the order of individual elements. By setting the default order (in run_setup) to 1 and setting the order of the sextupoles to 2 (using the ORDER parameter of the elements), one obtains symplectic tracking. Second, one may use the canonical variants of the individual elements. This involves modification of the lattice file, but allows retaining nonlinearities in dipoles and quadrupoles. The canonical elements are CSBEND, KQUAD, KSEXT, and MULT, where the latter is a general multipole. The FMULT element permits simulation of a multiple specified as a list of component strengths in an SDDS file. These elements also support classical synchrotron radiation energy losses.

elegant supports rf cavities with exact time dependence. These include the RFCA element, which simulates a basic rf accelerating cavity and the RFDF element, which simulates an rf deflector. The TWLA element simulates a traveling-wave linear accelerator, which is preferred for low-energy beams.

elegant supports a number of time-dependent ele-

ments. The BUMPER element permits simulation of a bumper (or kicker) magnet with a time-dependent waveshape specified via an SDDS file. The MODRF element provides simulation of an rf cavity with AM and PM modulation of the phase. RAMPRF provides simulation of an rf cavity with voltage, phase, and frequency waveforms from an SDDS file. RAMPP provides ramping of the central momentum in a simulation fashion. Together, these elements provide simulation of ramped machines, such as booster synchrotrons.

Several elements provide simulation of collective effects. The CHARGE element is used to impart charge to the beam. Having this as an element in the beamline allows elegant to vary the charge or assign random errors to it. The WAKE and TRWAKE elements provide Green-function-based simulation of longitudinal and transverse wakes, while the RFCW element combines simulation of an rf cavity with longitudinal and transverse wakes. The ZLONGIT and ZTRANVERSE provide simulation of impedances specified as tables of real and imaginary components as a function of frequency, or using a broad-band model. For multipass effects, the RFMODE and TRFMODE elements simulate resonator impedances with specified frequency and Q.

The CSRCSBEND and CSRDRIFT elements allow simulation of coherent synchrotron radiation (CSR) effects on the beam. The method uses a line-charge approximation [5]. It does not assume steady-state CSR nor does it assume a gaussian time distribution.

For storage rings, elegant simulates intra-beam scattering using the IBSCATTER element. A number of other elements also provide for beam scattering or excitation. The MATTER element simulates scatter and energy loss due to material in the beam path. The SCATTER element provides general scattering under user control, while the SREFFECTS element allows simulation of quantum excitation and damping effects in storage rings.

Several elements provide apertures of various types. The ECOL and RCOL elements provide elliptical and rectangular collimators. The MAXAMP element permits defining an elliptical beam tube that is valid for all following elements (until the next MAXAMP element). The SCRAPER element provides a single-jaw, straight-edge scraper that may be inserted from either side, from the top, or the bottom. The alpha magnet element incorporates its own scraper controls. The PFILTER element provides momentum filtration that is very convenient for removing high- and low-energy tails.

Most elegant elements have misalignment and tilt controls as part of the element definition. In addition, the MALIGN and ROTATE elements provide for misalignment and rotation of the beam. The CENTER element provides for centering specified beam coordinates (e.g., x or y). The MAGNIFY element provides for multiplication of particle coordinates by user-specified factors, which is useful if not particularly physical. The REMCOR element allows removing linear correlations among particle coordinates, which can be used to simulate certain types of corrections (e.g., residual dispersion after a bunch compressor) without having to perform them in detail.

elegant provides a number of elements for beam diagnostics. The HMON, VMON, and MONI elements are beam position monitors (BPM); the user may supply equations giving the BPM readout as a function of the actual x and y position in the device. The HISTOGRAM element provides SDDS output of histograms of transverse and longitudinal data. The WATCH element has several modes that result in output of beam data to an SDDS file; the user may choose particle coordinates; beam centroids; beam centroids and higher moments; or FFTs of turn-by-turn data (for storage rings).

5 OUTPUT FILES

elegant produces a large number of SDDS output files, but only when and as requested by the user. This prevents generation of large amounts of unneeded or perhaps meaningless information. At the same time, users may be confused about how to obtain certain output. As noted previously, many of the individual commands result in production of output files related to the computations they perform. In such cases, the user may give the command a filename to use for each type of output generated by the command. For example, the twiss_output command not only controls computation of Twiss parameters for internal use, but also allows the user to request that Twiss parameters be written to a file.

Many types of output are requested from the run_setup command. In general, any output that does not require special parameters is requested via run_setup. Also, any output that may result from several different action commands (e.g., tracking or optimization) is requested via run_setup.

Finally, a few accelerator elements produce output as well. In this case, the user specifies the name of the output file in the definition of the element. Table 1 summarizes the commands and elements that produce output files.

In order to make it easier for the user to generate names for output files without constant editing of the command and lattice files, elegant supports the concept of "incomplete" filenames for output files. The user specifies an incomplete filename by including the sequence "%s" in the filename. elegant detects this and substitutes the "rootname" for the simulation run. The rootname is derived automatically from the name of the command input file by removing the extension, or it may be specified explicitly in the run_setup command. This is discussed further in the manual.

6 SDDS TOOLKIT AND ELEGANT

As mentioned above, the SDDS Toolkit is the sole postprocessor for elegant and a number of other physics codes. This suite of programs provides general-purpose data analysis and display that can be used directly from the commandline or from within scripts prepared by the user. The programs can also be called from within elegant using

the `subprocess` command and the command-substitution syntax, as discussed above.

Users concerned about the stability of the Toolkit and the long-term accessibility of SDDS data may be reassured to know that SDDS is also a critical part of the APS control system. In fact, SDDS was originally developed for this purpose. Starting with commissioning in 1994, the vast majority of accelerator data was collected in SDDS files. The data logging system, orbit control system, configuration management system, and other vital systems all use SDDS files and Toolkit programs.

In this section, I will review some of the most-used SDDS Toolkit programs and give an indication of the application of each to `elegant` simulations. Because the Toolkit is based on the concept of self-describing data, each of the programs may be used with any of the data files described in the last section. However, there are some combinations that are used frequently and it is helpful to the new user to review these. Detailed syntax for using these programs is available in the manual [8]. In addition, all programs return a usage message if executed without arguments.

Like `elegant` itself, the SDDS Toolkit is not based on a graphical user interface (GUI), for the same reason. We have found a scripting, command-oriented environment more productive and better suited to the needs of research than the confines of a GUI environment.

6.1 Structure of SDDS Files

It will help to review briefly the structure of an SDDS file. An SDDS file begins with a header that describes the data in the file. Essentially, the header describes a complex data structure. Following the header are zero or more instances of this structure. Each instance is referred to as a "data page" or "page."

The header defines three types of entities: parameters, columns, and arrays. Each defined entity may have one of six data types: short integer, long integer, single-precision floating point, double-precision floating point, character, and character string. Parameters are scalar values. Columns are vector values that form a single table; that is, all the vectors have the same length and corresponding entries in the vectors form rows of data. Arrays are arbitrary, multidimensional entities and are the most flexible form. However, arrays are usually unnecessary and few applications use them.

The example of storing Twiss parameters and related data may make this clearer. In the output file from the `twiss_output` command, `elegant` uses parameters to store overall properties of a lattice, such as the tunes, chromaticities, equilibrium emittance, and momentum compaction factor. Not all parameters appear in all Twiss output files all the time; `elegant` "knows" which data is meaningful or valid and only puts that data in the file. `elegant` uses columns to store Twiss parameters as a function of s, along with element names and apertures.

In some cases, multiple pages of Twiss parameter data will be generated. For example, the user may invoke random errors and request output of the Twiss data for each set of errors, or the user may vary some magnet strengths and request Twiss data for each case.

If the user requests other data besides Twiss parameters, it will in most cases have the same page structure. For example, if transport matrix output is requested, each page of the matrix output file will correspond to the same page of the Twiss parameter file. The only guaranteed exceptions are the corrector and orbit output files from the `correct` command; these require multiple pages for each case because they must provide data before, during, and after correction. Even in these cases, the user can employ the `sddsprocess` program (see the next section) to remove unwanted pages and restore a one-to-one correspondence with other output files.

6.2 Commonly-Used Toolkit Programs

`sddsplot` is without a doubt the most-used Toolkit program. Some typical uses include scatter plots of particle phase space data, Twiss parameters vs. position, matrix elements vs. position, optimization progress vs. step number, turn-by-turn phase-space movies, and so on. `sddsplot` is used not only to display data directly from `elegant`, but also to display the results of processing with other Toolkit programs. In most cases where data is processed with the Toolkit, it ends up being displayed with `sddsplot`.

`sddsprintout` is another popular means of displaying data. Unlike other programs, `elegant` does not directly generate printouts of data. Doing so is not only inefficient in terms of disk space, but the printouts often do not satisfy the users' needs, containing insufficient accuracy, uninteresting data, or the wrong data. Instead, `sddsprintout` is used to create customized printouts from any SDDS file. The user can thus see only the data that is interesting, to the required precision, and in a specified order. Like `sddsplot`, `sddsprintout` is frequently found at the end of a chain of SDDS processing commands, but can be used directly on the files generated by `elegant`.

`sddsprocess` is a general-purpose data processing and filtering utility. It performs statistics on column data and places the results in parameters. So, for example, one can use it to compute the average of the horizontal and vertical beta functions, or the maximum dispersion. It also creates new columns and parameters based on user-supplied equations. Hence, one could use `sddsprocess` to make a new column containing $\beta_x^{\frac{3}{2}}$ and then a new parameter $\langle\beta_x^{\frac{3}{2}}\rangle$. In addition to accepting equations, `sddsprocess` accepts *equation templates*, so that one can process many similar columns or parameters in a similar fashion. Taking statistics of multiple columns is similarly easy using wildcards to select the columns of interest. Filtering of data can be performed based on numerical values or string values. For example, one could select all quadrupoles from a Twiss file and compute the average beta functions at those locations only.

Some other frequently-used tools are:

- `sddshist` and `sddsmultihist` for histograms.
- `sddsenvelope` for finding maximum beta functions, beta beats, and so on, over many configurations.
- `sddsfft` for frequency domain analysis.
- `sddssmooth` for smoothing data.
- `sddspfit`, `sddsexpfit`, and `sddsgfit` for polynomial, exponential, and gaussian fits, respectively.
- `sddscollapse` is used to collapse a file containing many pages with parameters and columns to a file containing a single page. This single page contains one row for each page in the original files. The former parameters become columns in the new file. `sddscollapse` is commonly used to "throw away" detailed data after analysis has been performed.
- `sddsxref` adds selected parameters, columns, and arrays from one file to another file. It can line up rows in the files by comparing user-specified columns. It could be used, for example, to bring data from the Twiss parameter file into a file containing closed orbit data.
- `csv2sdds` converts comma-separated-value data to SDDS. `plaindata2sdds` converts unadorned text or binary data to SDDS; `sdds2plaindata` performs the reverse conversion.
- `sddsquery` provides a printout showing what parameters, columns, and arrays are in an SDDS file. The printout includes data types and units.

7 EXAMPLES

In this section, I give several examples of the application of `elegant` to real-world problems. My intention is to show that `elegant` can be applied to some very complex problems and that it is very useful when designing and upgrading accelerators. I will not present these examples in great detail, since this is not a tutorial. Instead, I will summarize how `elegant` was used in each case so that the reader can judge the capabilities of the program.

Like many projects that use `elegant` and other SDDS-compliant simulation codes, most of these projects made use of a multi-workstation queue [9] utilizing up to 50 Sun workstations. Because scripts are used for setting up, submitting, and postprocessing jobs, it is possible to run many jobs simultaneously for greatly improved productivity. The desire to use this kind of computing environment is one of the reasons that `elegant` is not GUI-based. The GUI model tends to assume a single user in front of a single computer.

7.1 APS Positron Accumulator Ring Design

One of the first accelerators designed using `elegant` was the Positron Accumulator Ring (PAR) [10, 11, 12] for the APS. The PAR is a small ring with a 30.7 m circumference and eight, 45-degree dipole magnets having a bending radius of 1.02 m. Other magnets include 16 quadrupoles in 4 families, and 6 sextupoles in 2 families. The PAR has two rf systems, a first-harmonic system for capturing the beam, and a twelfth-harmonic system for compressing the bunch length. In many ways, the PAR is more difficult to model than a third-generation light source, as the latter generally has large bending radius, single rf systems, single-turn kickers, and relatively quick damping.

Matching for the PAR was done using MAD, while tracking and other analyses were performed with `elegant`. Some of the simulations performed with `elegant` include:

- Simulation of injection and extraction processes using measured waveforms for the kickers. Since the kicker pulses are longer than a single turn, simulation was needed to ensure that the partially-damped stored beam was not lost during injection. Extraction involves three kickers, two of which form a closed bump, making for a multi-turn extraction. `elegant` was used to find the optimum kicker strength for extraction.
- Simulation of final bunch purity for various ramp-up profiles of the twelfth-harmonic cavity. These simulations included radiation damping and excitation.
- Simulation of beam stability in the presence of detuned harmonic cavity. This was used to specify the required detuning of the harmonic cavity when unpowered. As predicted, the PAR was found to exhibit a sawtooth instability prior to detuning of the harmonic cavity.
- Simulation of dynamic aperture as a function of momentum and in the presence of random and systematic strength, multipole, and alignment errors.
- Testing of diagnostics placement in the transport lines leading to and from the PAR, and testing of injection in the presence of errors in the transport lines.

7.2 APS Dog-leg Lattices

The APS has 40 straight sections, each about 5 m long. Originally, it was thought that users would desire long undulators, but experience has shown that most individual users are satisfied with an undulator that requires only half the available space. This means that half the space is unused. Recently, a proposal was made to install three-magnet bumps in one or more straight sections, such that two insertion devices could occupy the straight section, with a separation angle of about 1 degree between the beamlines. This is known as an "ID Dog-Leg."

Concerns about the dog-leg idea included whether the emittance would be spoiled by one or more such insertions, how much the ideal orbit length would change, and the usual concerns about dynamic aperture. `elegant` was used to match a series of dog-leg configurations for separation angles of up to 1 degree, with emittances from 8 nm to about 3.5 nm. The matching was highly automated and performed in stages, starting with no separation and working up to the maximum separation. Splicing of dog-leg and non-dog-leg cells was also performed to verify that the emittance and dynamic aperture did not suffer.

Because `elegant` uses SDDS files, automation of the process was relatively easy. Scripts were written to set up a series of runs for different emittances and gradually increasing separation angle. Because `elegant` can directly perform matching on quantities like the equilibrium emittance, it was easy to obtain lattices that had the desired emittance for the dog-leg, non-dog-leg, and transitional cells.

`elegant` was also used to determine the change in the length of the central orbit, which involved simulating several girder rotations and displacements. The program was further used to explore alternate configurations that reduced the path-length change and others that involve distortion of the sectors around the dog-leg to eliminate pollution of x-ray BPMs by bending magnet radiation [13].

7.3 APS Top-Up Safety Tracking

"Top-up" [14] refers to a new mode of operating a synchrotron radiation source. Traditionally, synchrotron radiation sources are filled with shutters closed, then shutters are opened (giving light to the users) while the beam decays over many hours. In top-up mode, injection occurs frequently with shutters open. A concern with top-up mode is whether injected beam might, due to some equipment fault, exit a user beamline, resulting in serious injury. Clearly, if a dipole magnet were to fail, this is physically possible. In order to prevent such an accident without interlocking every dipole, APS was interested in simply interlocking on the stored beam, on the assumption that if there was stored beam, then the dipoles must all be operating properly. Analytical methods [15] showed that this was plausible. The goal of top-up safety tracking [16] was to provide greater certainty that such an interlock was sufficient.

Top-up safety depends on having apertures in the ring that limit the possible excursion of the stored beam. Hence, `elegant`'s ability to have various types of apertures was important. For top-up safety tracking, about 500 runs are required for each aperture configuration. Runs are grouped according to whether they simulate stored beam or backward-tracked injected beam, and according to the "failure scenario."

A failure scenario always involves a single dipole that is either fully or partially shorted. In most cases, it also involves another assumed failure, such as another nearby magnet that is adjusted so as to make an accident most likely. For example, if a corrector downstream of the shorted dipole is driven to maximum current, it might correct the orbit for the stored beam, thus fooling the interlock. While this is improbable, we felt it necessary to explore such possibilities in order to ensure that an accident could not occur.

For each scenario type, a Tcl/Tk script is used to set up and submit the simulation runs. This script is itself usually invoked by another script that starts all the runs involved in a particular aperture configuration. These scripts greatly simplify the task of setting up and running a new round of simulations.

For each scenario, a specific script is used to postprocess the data and produce a simple results file (again, an SDDS file). These scripts also detect problems (e.g., missing data that might result from a workstation crashing), and to prevent using bad data, any simulations with problem data are deleted and must be run again. The user can easily do this by reinvoking the submission script. Like startup, postprocessing can be invoked with a single command. This command executes the scenario-specific scripts, then collates the scenario-specific results files into a single result file. In addition, the script produces a single value showing whether the configuration is unsafe.

Both the startup and postprocessing scripts use the SDDS Toolkit for data preparation and analysis. In addition to using SDDS files for all output, `elegant` uses SDDS files for configuration of tracking and for tracking input. Most of these files are prepared automatically by the scripts or by other `elegant` runs (a few represent external input, e.g., the apertures, and are prepared manually). Further, different aspects of the same scenario sometimes share data. Because data is passed between simulations using SDDS files, there is no risk of transcription error.

7.4 APS Bunch Compressor Design and Tolerance Analysis

The APS has an FEL project known as the Low Energy Undulator Test Line (LEUTL) [17, 18]. In order to push this FEL to saturation with fewer undulators and at shorter wavelengths, we embarked on a rapid program to build a bunch compressor for the APS linac [19]. It was desired to have a system that permitted variation of the R_{56}, which would be achieved by moving the magnets, rather than having very wide magnets. In addition, we desired a system with variable symmetry, to test the hypothesis that such a system has smaller CSR-induced emittance growth. All of the matching and simulation for this project was done with `elegant` [20].

The optics of the APS linac were originally designed for creation and capture of positrons. A first step in the bunch compressor project was a new optics configuration for the linac, which involved rearranging the existing quadrupoles and performing matching for the thermionic and photocathode guns. This new configuration made the linac bunch-compressor-ready, in addition to improving oper-

ations prior to bunch compressor installation. Response matrix measurements indicated good agreement between `elegant` and the reconfigured linac.

Because the bunch compressor chicane is flexible and because we have multiple sources of beam, it was necessary to perform matching for many configurations. This was required to ensure that we could accomodate various R_{56} values, beam sources, and acceleration profiles within reasonable limits for power supplies.

The starting point of the simulations was matching of about 80 chicane configurations for a grid of R_{56} and asymmetry values. This matching included matching of dispersion and floor coordinates, to ensure that the various configurations were physically compatible.

Next, longitudinal matching was performed for selected configurations for the photocathode gun, to obtain desired beam currents, energy profiles, and minimal energy spread at the end of the linac. The longitudinal dynamics in the linac are sensitive to the initial distribution and wake fields, so that matching had to be performed by tracking a beam of macro-particles.

Following this, beta function matching was performed for each configuration. This matching started from a "hand matched" configuration, working down the linac in four stages. Data transfer from the chicane configuration, to longitudinal matching, to transverse matching, was performed automatically using SDDS files and scripts. Once the scripts and input file templates were prepared for these runs, any number of configurations could be explored with little additional work. This was very important given the rapid nature of the project, since it allowed quick evaluation of proposed changes.

After the matching was completed, tolerance simulations were performed for all of the configurations. This started with parameter "sweeps," wherein a single accelerator parameter (e.g., an rf phase) was swept over a range to determine its impact on important beam properties (e.g., bunch length, gain length). Once all the parameter sweeps were completed, the results were analyzed to determine which configurations were least sensitive to errors, and what the tolerances were. Finally, all-inclusive random-error simulations were performed for those configurations, confirming the tolerance determination and assessing the impact of relaxed tolerances.

8 ACKNOWLEDGEMENTS

I developed the early versions of `elegant` while working at the Stanford Synchrotron Radiation Laboratory under Helmut Wiedemann. Since then, `elegant` has grown and improved dramatically, thanks in no small part to bugs found and suggestions made by users, including Paul Emma, Louis Emery, Zhirong Huang, Eliane Lessner, John Lewellen, Steve Milton, and Nick Sereno. I'm grateful to John Galayda for the opportunity to work on `elegant` and SDDS while at APS.

9 REFERENCES

[1] M. Borland, "A Self-Describing File Protocol for Simulation Integration and Shared Postprocessors," Proc. of the 1995 PAC, Dallas, Texas, pp 2184-2186 (1996).

[2] M. Borland, "A Universal Postprocessing Toolkit for Accelerator Simulation and Data Analysis", Proc. of the 1998 ICAP, Monterey, California, to be published.

[3] M. Borland, "User's Manual for `elegant`," available on-line at http://www.aps.anl.gov/asd/oag/manuals/ elegant_ver14.1/elegant.html.

[4] H. Grote and F. C. Iselin, "The MAD Program (Methodical Accelerator Design)," CERN/SL/90-13(AP), 1991, Geneva, Switzerland.

[5] E. L. Saldin, E. A. Schneidmillter, and M. V. Yurkov, "On the coherent radiation of an electron bunch moving in an arc of a circle," NIM A 398 (1997) 392.

[6] M. Xie, "Design Optimization for an X-Ray Free Electron Laser Driven by SLAC Linac," Proc. 1995 PAC, Dallas, May 1-5, 183.

[7] M. Borland, "A High-Brightness Thermionic Microwave Electron Gun," SLAC Report 402, Chapter 3, February 1991, Ph. D. thesis.

[8] M. Borland, "Users Guide for SDDS Toolkit," available on-line at http://www.aps.anl.gov/asd/oag/manuals/ SDDStoolkit/SDDStoolkit.html.

[9] T. P. Green, "Research Toward a Heterogeneous Networked Computer Cluster: The Distributed Queuing System Version 3.0," SCRI Technical Publication, 1994.

[10] M. Borland, "Commissioning of the Argonne Positron Accumulator Ring," Proc. of the 1995 PAC, May 1-5, 1995, Dallas, Texas.

[11] M. Borland, "Construction and Commissioning of the Positron Accumulator Ring for the APS," Proc. of the 1994 Conference on Applications of Accelerators in Research and Industry, NIM.

[12] M. Borland, "Update on the Argonne Positron Accumulator Ring", Proc. of the 1993 PAC, Washington, DC, May 1993.

[13] G. Decker, O. Singh, "A Method for Reducing X-ray Background Signals from Insertion Device X-ray Beam Position Monitors," Phys. Rev. ST Accel. Beams, **2**, 112801 (1999).

[14] L. Emery, M. Borland, "Top-Up Operation Experience at the Advanced Photon Source," Proc. of 1999 PAC, March 29-April 2, New York, 200-202.

[15] L. Emery, M. Borland, "Analytical Studies of Top-Up Safety for the Advanced Photon Source," Proc. of 1999 PAC, March 29-April 2, New York, 2939-2941.

[16] M. Borland, L. Emery, "Tracking Studies of Top-Up Safety for the Advanced Photon Source," Proc. of 1999 PAC, March 29-April 2, New York, 2319-2321.

[17] S.V. Milton et al., "The FEL Development at the Advanced Photon Source," Proc. FEL Challenges II, SPIE, January 1999, to be published.

[18] S. V. Milton et al., "Observation of Self-Amplified Spontaneous Emission and Exponential Growth at 530 nm," Phys. Rev. Let., **85**(5), 988-991.

[19] M. Borland et al., "A Highly Flexible Bunch Compressor for the APS LEUTL FEL," Proc. 2000 LINAC Conference, Monterey, to be published.

[20] M. Borland, "Design and Performance Simulations of the Bunch Compressor for the APS LEUTL FEL," Proc. 2000 LINAC Conference, Monterey, to be published.

[21] J. W. Lewellen, et al., "A Hot-Spare Injector for the APS Linac," Proc. of 1999 PAC, March 29-April 2, New York, 1979-1981.

Table 1: Output files created by elegant and command used to obtain them

output	command	parameter
amplification factors	amplification_factors	several
inferred linear matrix from tracking	analyze_map	output
closed orbit	closed_orbit	output
closed orbit or trajectory before and after correction	correct	trajectory_output
orbit corrector values before and after correction	correct	corrector_output
beam/corrector statistics before and after correction	correct	statistics
orbit/trajectory response matrix	correction_matrix_output	response, inverse
random error values for elements	error_control	error_log
dynamic aperture	find_aperture	output
dynamic aperture search boundary	find_aperture	boundary
floor coordinates	floor_coordinates	filename
transport matrices	matrix_output	SDDS_output, printout
optimization log (text)	optimization_setup	log_file
element dictionary (text)	print_dictionary	filename
final particle coordinates	run_setup	output
centroids vs. s from tracking	run_setup	centroid
sigma matrix etc. vs. s from tracking	run_setup	sigma
final particle and accelerator properties	run_setup	final
initial coordinates of transmitted particles	run_setup	acceptance
coordinates of lost particles	run_setup	losses
magnet layout	run_setup	magnets
lattice parameters	run_setup	parameters
SASE FEL computations	sasefel	output
lattice file (text)	save_lattice	filename
Twiss parameters	twiss_output	filename
particle coordinates at interior points	WATCH element	FILENAME
beam centroids at interior points	WATCH element	FILENAME
beam sigmas at interior points	WATCH element	FILENAME
beam histograms at interior points	HISTOGRAM element	FILENAME
CSR wakefields	CSRCSBEND element	OUTPUT_FILE