SDDS-BASED SOFTWARE TOOLS FOR ACCELERATOR DESIGN*

M. Borland, L. Emery†, H. Shang, R. Soliday, ANL, Argonne, IL 60439, USA

Abstract

The Self-Describing Data Set (SDDS) file protocol is a standardized way to store and access data and is the basis of an extensive toolkit. It is also the file protocol used for many accelerator design tools. Over the years, several of these SDDS-compliant accelerator programs (e.g., `clinchor`, `elegant`, `estat`, `shower`, and `spiffe`) have been developed at the Advanced Photon Source. Also, existing accelerator design tools for which the source code is available (e.g., ABCI, GENESIS, GINGER, MAFIA, and URMEL) have been converted to read and write SDDS files. As a result, we now have a capable set of accelerator codes that make use of the same data format and the same pre- and postprocessing suite. Further, the SDDS toolkit program `sddsoptimize` can be used around any of these tools or around a script that runs one or more of these tools. This provides the capability of very general, multicode optimization. In this paper, we discuss the capabilities of the existing SDDS-compliant accelerator codes, then provide examples of applications of these tools.

INTRODUCTION

The accelerator field is well supplied with simulation codes. Accelerator designers frequently need to make use of codes by many authors, developed in different programming languages. Although this need is common, it is usually not easy to integrate the results of these codes. Part of the reason for this is that codes do not share a common, stable data protocol. Instead, each code (or group of codes) has a custom data protocol that may change from one version to the next. Another reason is that each code has its own pre- and postprocessor, which typically don't recognize data from other codes.

This means that integration of two or more codes requires writing a custom translation program. Such translators tend to be fragile as the developers of the simulation code don't support them and may change their protocols without notice in new code versions. This fragility is largely due to the widespread failure to use robust self-describing data protocols.

At APS, we developed a common self-describing data standard [1] for used by codes and controls systems. This standard is called SDDS, for Self-Describing Data Sets. SDDS datasets, very briefly, consist of a header that describes the contents of the file, including names, data types, and units. The header describes a data structure containing parameters, a table, and arbitrarily-dimensioned arrays. Access to data in SDDS files is *by name only*. This is the key feature for a robust protocol.

In addition to requiring accelerator codes to read and write SDDS files, we created a suite of data processing and display tools that work with SDDS files. In effect, we created a common pre- and postprocessing toolkit that is used by our codes and codes we have modified. This set of approximately 80 generic programs is referred to as the SDDS Toolkit [1].

As indicated, a major advantage of using SDDS files is that data from any code can be used with any other code. This is robust due to the use of SDDS files, meaning that one code can be upgraded without requiring a change of the other code. In addition, with SDDS it is straightforward to process and display data from several codes together. The SDDS Toolkit also provides the ability to make transformations of data, which is useful when codes have different conventions (e.g., for phase-space quantities). Finally, using SDDS means that adding capabilities to a simulation code is faster and easier. The new data is simply placed in SDDS files where it can be accessed with the existing suite of tools.

In addition to the SDDS Toolkit, users can import SDDS data directly into programming environments like C/C++, FORTRAN, IDL, Java, MATLAB, and Tcl/Tk, using libraries created and supported by APS. These libraries, like the rest of the SDDS software and our simulation codes, are covered by an Open Source license and available for download from our web site. The codes discussed are all available for UNIX environments, including LINUX, Solaris, and MAC OS-X, and also (usually) for Windows.

SDDS-COMPLIANT CODES AND THEIR CAPABILITIES

In this section, we briefly review the capabilities of some of the existing SDDS-compliant codes. We also attempt to briefly indicate how these codes can be used together. Detailed examples of this will appear in the next section.

General Accelerator Simulation with `elegant`

The program `elegant` [2] was the first of the SDDS-compliant accelerator codes. Because it performs general-purpose accelerator simulation, it is at the center of the SDDS-compliant code set. `elegant` performs optics calculations, errors and lattice correction, various types of particle tracking, and many other functions. Multidimensional scans may also be performed.

In addition, `elegant` provides a general-purpose optimization capability that may be unique: the user-defined

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

† emery@aps.anl.gov

penalty function may include the usual quantities such as Twiss parameters and matrix elements (at any point in the system), but also quantities like the equilibrium emittance or momentum compaction. In addition, properties (e.g., centroids, beam sizes) of tracked particle distributions may be optimized.

Besides the typical beamline optical components, `elegant` provides some unusual elements. These simulate various collective effects, such as longitudinal and transverse short-range wakefields (or impedances), longitudinal and transverse resonator impedances, coherent synchrotron radiation, and intrabeam scattering. There are also time-dependent elements such as rf cavities, rf deflectors, kickers, and single-bunch digital feedback.

The SCRIPT element allows the user to add a custom-defined element for tracking in `elegant` by specifying an external program that is used to transform the beam distribution. The program must be SDDS-compliant. For noncompliant programs, a wrapper script must be written to translate between SDDS and the program's unique data format. Such scripts are easily written using the SDDS Toolkit, which features several programs (`sddsprintout`, `sdds2stream`, and `sdds2plaindata`) designed to convert SDDS data into plain ASCII or binary data.

Specialized Tools for Use with `elegant`

As mentioned above, the generic SDDS Toolkit can be used to process and display data from SDDS files produced by `elegant` or any other SDDS-compliant code. In addition, there are some specialized SDDS tools that are designed to support or work with `elegant`. Of course, they can be used with other SDDS-compliant codes as well.

`ibsEmittance` — Computes intrabeam scattering (IBS) growth rates using Twiss parameter data from `elegant`. The algorithm is based on an improved version of IBS code in the program ZAP [3]. `ibsEmittance` will also compute transverse and longitudinal emittance evolution by integrating the differential equations for the emittances. Growth rates are recomputed as the emittances change.

`sddsanalyzebeam` — Analyzes particle distributions produced by `elegant` or other codes that use the same naming convention. Output includes Twiss parameters, emittance, and beam sizes. Output can be used as input to Twiss parameter propagation in `elegant`.

`sddsbrightness` — Computes undulator brightness curves using Twiss parameter and emittance data from `elegant`. Supports several methods, from very simple estimates to full-blown calculations based on the program `xop` by Dejus [4].

`sddsemitmeas` — Analyzes quadrupole scan data to find the sigma matrix of a beam. `elegant` is used to compute the transfer matrix of a beamline as one or more quadrupoles are varied. `sddsemitmeas` uses this data along with beam size data from simulation or experiment to determine the sigma matrix. Error analysis is included.

`sddsmatchtwiss` – Performs phase-space transformations of particle distributions. For example, the beta functions can be changed to match a beam into a simulation even if the matching optics haven't been developed yet. `sddsmatchtwiss` will also take output of `elegant` Twiss computations or `sddsanalyzebeam` computations to specify which Twiss parameters to match to the beam.

`sddssasefel` — Performs computations of self-amplified spontaneous emission free-electron lasers using the method of M. Xie [5]. The beam properties can be those computed by an `elegant` tracking run or prepared in some other fashion. Xie's method is also used internally to `elegant`, but `sddssasefel` has additional features such as optimization of parameters that are not specified in the input data.

`sddsrandmult` — Prepares data giving the multipole content of quadrupoles or sextupoles in the presence of various construction errors. This data is accepted by `elegant` for tracking (e.g., dynamic apertures).

Free-Electron Laser Simulation

As mentioned above, `elegant` performs a simple FEL calculation using M. Xie's parametrization, which gives a good estimate of FEL behavior. For more exact results, or to perform start-to-end simulations [6], the user needs to use an FEL code such as GENESIS [7] or GINGER [8]. At APS, we have modified GENESIS [9] to read and write SDDS files, making it easy to evaluate FEL performance for a given particle distribution from `elegant`. To do this, we made use of the existing BEAMFILE feature of GENESIS, which allows specifying centroid and rms properties of a series of independent beam slices. The necessary slice analysis is performed with the program `elegant2genesis` [9]. GENESIS can then simulate each slice in turn, producing SDDS files with radiation properties for each slice along the undulator. As discussed in more detail in [6], this data is readily associated with the input slice properties and the settings or errors in the accelerator, making quantitative understanding of slice-to-slice output variations possible. This is made easy by the fact that all the input and output data is in SDDS files. An SDDS-compliant version of GINGER has also been produced [10].

Other Codes

ABCI/APS — Our version of ABCI [11] produces an SDDS file giving the longitudinal or transverse wakes. These wakes can be used directly with the WAKE and TRWAKE elements, respectively, in `elegant`. These elements perform simulation of beam interaction with single-pass wakefields. If necessary, the output from ABCI can be processed with the SDDS Toolkit before using the data with `elegant`. This might be necessary, for example, to deconvolve the effects of nonzero bunch length in ABCI.

`clinchor` [12] — This program simulates single- or coupled-bunch instabilities driven by resonant modes. The mode properties are taken from the SDDS output file generated by our version of URMEL (see below). Lattice information is taken from the elegant Twiss parameter output file.

estat [13] — This is a simple 2-dimensional electrostatic solver. Output from estat can be used in tracking with the BMAPXY element in elegant. This is a good example of a fairly small, simple code that becomes much more useful by virtue of the SDDS Toolkit (which obviates the need for a postprocessor) and interaction with elegant.

MAFIA/APS — We have modified the MAFIA version 2.04 [14] T3 calculator to write SDDS output of wake potentials. Program mafiaTimeData converts the T3 internal format file into several SDDS files of wakefield and electromagnetic field data. The field data from the frequency-domain solver are converted to SDDS by program mafia2sdds.

shower [15] — We developed an EGS4 [16] wrapper program called shower that simplifies use of the EGS4 electron-gamma shower simulation code. Instead of writing MORTRAN macros, the user specifies the geometry using a simple text file. shower accepts input and output particle distributions in SDDS files. Hence, one can easily simulate interaction of an accelerator beam with matter and the subsequent behavior of shower products in the accelerator. Postprocessing to obtain dose rates is also straightforward with the SDDS Toolkit.

spiffe [13] — This is a 2.5-dimensional particle-in-cell code written at APS, intended for rf gun design. spiffe produces SDDS particle output files that are read directly by elegant. spiffe will use field profiles generated by URMEL/APS to impose cavity fields on the beam.

URMEL/APS — Our version of URMEL produces an SDDS file giving the on-axis field profiles for all modes. This data can be used with spiffe to simulate the accelerating mode of an rf gun, for example. URMEL/APS also produces a file giving the frequency, shunt impedance, and Q for each mode. This file can be used with the FRFMODE and FTRFMODE elements in elegant, which simulate longitudinal and transverse resonant cavity modes.

APPLICATION EXAMPLES

Here we briefly mention several applications of the above suite of simulation codes.

- elegant and shower were used to simulate beam losses and radiation production in the APS injector and ring, for evaluation of shielding design and radiation dose to undulators [15].
- shower was used as a part of an elegant beamline using the SCRIPT element for evaluation of a beam collimation system for the APS booster-to-storage-ring transport line.
- spiffe and elegant were used to simulate the APS thermionic rf guns, transport lines, and subsequent acceleration. These simulations helped to solve beam transport problems with one of the rf guns that had prevented use of the gun for top-up.
- PARMELA, elegant, sddsmatchtwiss, elegant2genesis, and GENESIS were used for start-to-end jitter simulation [6] of the Linac Coherent Light Souce [17]. PARMELA [18] and elegant were used to simulate the APS photoinjector.
- URMEL and clinchor were used to evaluate staggering of APS resonant cavity modes to avoid multi-bunch instabilities [12]. This work was done prior to the development of the SDDS system, but the codes have been made SDDS-compliant and are used for similar computations.
- MAFIA and elegant are being used for simulation of the bursting mode instability in the APS [19].
- elegant and sddsemitmeas are used to analyze emittance measurement quadrupole scans for APS linac experiments. The experiments are performed and analyzed by a Tcl/Tk script. Data collection is performed with the SDDS-compliant EPICS Toolkit [20].
- elegant and sddsbrightness are used to provide brightness curves for APS users, updated every 15 minutes.

REFERENCES

[1] M. Borland, “A Self-Describing File Protocol for Simulation Integration and Shared Postprocessors,” 1995 PAC, Dallas, Texas, 2184 (1996).

[2] M. Borland, “elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation,” Advanced Photon Source LS-287, September 2000.

[3] M. Zisman, PAC 1987, 991 (1988).

[4] M. Sánchez del Río and R. J. Dejus. “XOP: Recent developments” SPIE proceedings 3448 (1998).

[5] M. Xie, http://epaper.kek.jp/p95/ARTICLES/TPG/TPG10.PDF.

[6] M. Borland *et al.*, PAC 2001, Chicago, 2707 (2002).

[7] S. Reiche, NIM A 429, 242 (1999).

[8] W.M. Fawley, LBNL CBP Tech Note-104 (1995).

[9] Y. Chae and R. Soliday, PAC 2001, Chicago, 2710 (2001).

[10] W.M. Fawley, private communication.

[11] Y. Chin, PAC 1993, Washington D.C., 3416 (1994).

[12] L. Emery, PAC 1993, Washington D.C., 3360 (1994).

[13] M. Borland, private communication.

[14] R. Klatt and T. Weiland, SLAC Report 202 (1986), Version 2.04.

[15] L. Emery, 1995 PAC, Dallas, Texas, 2039 (1996).

[16] W.R. Nelson *et al.*, SLAC 265, 1985.

[17] “LCLS CDR,” SLAC-R-593, April 2002.

[18] J. Billen, Los Alamos National Laboratory report LA-UR-96-1835 (1996).

[19] Y. Chae, “The Impedance Database and its Application to the APS Storage Ring,” these proceedings.

[20] H. Shang *et al.*, “New Features in the SDDS-Compliant EPICS Toolkit,” these proceedings.