Commissioning Software Tools at the Advanced Photon Source*

L. Emery

Argonne National Laboratory

9700 So. Cass Ave., Argonne, IL 60439

Abstract

A software tool-oriented approach has been adopted in the commissioning of the Advanced Photon Source (APS) at Argonne National Laboratory, particularly in the commissioning of the Positron Accumulator Ring (PAR). The general philosophy is to decompose a complicated procedure involving measurement, data processing, and control into a series of simpler steps, each accomplished by a generic toolkit program. The implementation is greatly facilitated by adopting the SDDS [1] (self-describing data set) protocol, which comes with its own toolkit. The combined toolkit has made accelerator physics measurements easier. For instance, the measurement of the optical functions of the PAR and the beamlines connected to it have been largely automated. Complicated measurements are feasible with a combination of tools running independently.

I. Introduction

There generally occur two types of control room activities during commissioning of an accelerator. One is letting the beam "go" as far as it can, and tweaking all manner of possibly relevant "knobs" to improve performance. The other is carefully measuring the beam properties, optical functions, etc. in order to deduce sources of limitations, or simply to verify that the accelerator performance is within specification. Since beam studies time is highly valued, an efficient and flexible system of software tools for collecting, processing, and displaying of data was developed at APS. These tools have been in use since the beginning of the commissioning of the APS PAR, and have greatly aided accelerator physics personnel in the control room.

This paper briefly introduces the toolkit concept, then gives an overview of the accelerator physics tools written so far. Specific examples of applications of beam control and measurements are presented afterwards.

II. SDDS Toolkit

A toolkit is a set of generic and cooperative programs whose actions may be combined to produce a more complicated action. The motivation of implementing a toolkit is to eliminate the repitition of graphics and data processing programming work among computer codes. At APS, M. Borland had already developed the SDDS toolkit [1] for postprocessing and plotting output from many accelerator physics codes. This toolkit requires the use of the SDDS self-describing file protocol [1] for data files. Data files are organized in "pages" consisting primarily of tables of data. The most useful feature of the protocol is attributing names, units, and descriptions to the data by way of a file header. Thus columns in the data tables can be operated upon by listing their names as part of an option of a tool command line. For instance, suppose that an SDDS data file, `aps.twi`, contains the Twiss parameters of the APS ring

with columns `s`, `betax`, and `betay` defined at every element. Then the data can be plotted with the **sddsplot** tool: **sddsplot** `-col=s,(betax,betay)` `aps.twi`. Data from any SDDS data file can be plotted in a similar way. A consistent data file protocol is the common thread that makes it possible for the members of the toolkit to work together. Thus, to take advantage of the SDDS toolkit, our control system tools read and output SDDS protocol compliant data files.

If one didn't use a toolkit approach, one would typically end up with data and graphics postprocessors that work only for the output of one physics code. Examples of these situations are too numerous to list.

The SDDS toolkit will not be explained in detail in this paper, as the toolkit is too extensive to fit. However, some of the SDDS tools will be mentioned later.

III. EPICS Toolkit

The control system at APS is the Experimental Physics and Industrial Control System (EPICS) [2]. In EPICS, a database record is known as a process variable (PV), a term we'll generalize to mean any accelerator data obtained from EPICS.

Below is a list of EPICS tools in frequent use in beam studies:

- **burtrb**, **burtwb**, **burtmath** (Back-Up and Restore Tool): Save and restore accelerator configurations, respectively, in SDDS files.
- **sddsmonitor**, **sddsvmonitor**, **sddswmonitor**: Log PV data at regular intervals. Optionally, the conditions for a glitch or an oscilloscope-style trigger on a PV can be specified for the logging of data in a time window around those events. These tools are extremely useful for correlating data and diagnosing problems.
- **sddsexperiment**, **sddsvexperiment**: Vary PVs, and measure PVs with averaging. These tools require an input file of namelist-type commands describing the PVs to vary and measure. The results are written to a file which can, for example, be plotted directly using the **sddsplot** SDDS tool or be processed by one of the SDDS least-squares-fit tools.
- **sddscontrollaw**: Analogous to the term in control theory, this program uses a matrix and a list of PV readbacks (outputs) as a vector to calculate new setpoints of PVs (actuators). Suppose that R_i is a vector made from the list of readbacks at time index i, M is the correction matrix, and S_i is the vector of setpoints at time index i, then **sddscontrollaw** computes the new setpoints with $S_{i+1} = S_i - M R_i$. (The minus sign is present to indicate a correction.) The procedure can be repeated at a specified time interval.
- **squishPV**: Tweaks a list of PVs (e.g. correctors) to minimize the sum of another list of PV values (e.g. BPM readbacks). An SDDS input file provides the lists of PVs. The program tweaks one PV at a time, and leaves the tweaked PV at the value that gives the smallest sum for the readback PVs' absolute values. The procedure is naturally slower than **sddscontrollaw**'s which utilizes a correction matrix. However, the **squishPV** method is more robust in that it

works in the presence of gross readback offsets and errors, and in ignorance of response matrices.

These tools are very general. They don't even presume that an *accelerator* is being controlled. Scripts of command-line tools written to work for one accelerator at APS have been easily adapted to work with another.

The SDDS and EPICS toolkits are accessed through the command line and do not use graphical user interfaces (GUIs), and for good reason. In our experience, the alternate use of mouse operations and keyboard input, the hallmark of GUI, is an impediment rather than an advantage to accomplishing any serious work. In addition, sequences of command lines can be put into a script, which isn't possible with GUIs.

The usefulness of these tools emerges when a few applications are given in the next section.

IV. APPLICATIONS

A. Optical Function Measurements

In commissioning a beamline or storage ring, one verifies that the optics matches the model by making beam response measurements, that is, measuring the response of BPM readings as a function of steering magnet settings.

In the PAR, such measurements have been used to improve the optical model [3]. In the storage ring, a first turn trajectory beam response measurement has so far been used to identify grossly problematic quadrupole magnet supplies.

The beam response measurement starts with **sdds-experiment** with BPM readbacks as the measured PVs and a corrector power supply current setpoint as a variable PV. The slopes of the **sddsexperiment** output data—obtained with the tool **sddsslopes**—give the beam response.

The PAR measurement will be demonstrated as an example. The command which runs **sddsexperiment** for the PAR corrector P1H1 is "**sddsexperiment** P1H1.exp" where the command input file P1H1.exp is shown below.

```
&measurement    control_name = "P1P1:x",
        column_name="P1P1:x", units=mm
        number_to_average = 5,
&end
&measurement    control_name = "P1P2:x",
        column_name="P1P2:x", units=mm
        number_to_average = 5,
&end
... (repeat the above for the rest of BPMs)
! do first corrector in PAR
&variable control_name = "P1H1:CurrentAO",
        column_name="P1H1", units="A",
        relative_to_original=1,
        index_number = 0, index_limit = 5,
        initial_value = -2.0, final_value = 2.0,
&end
&execute
        outputfile = "P1H1.sdds"
        post_change_pause=3
        intermeasurement_pause=0.5
&end
```

The `measurement` command requires the fields control_name and column_name. The control_name field gives the name of the PV (i.e. P1P1:x) that is read. The name of the column_name field will be given to the measurement data in the output file. Having the flexibility of naming data columns is useful when a PV name is not very descriptive. The other fields are self-explanatory. The `variable` command shown here has four additional fields. They instruct the program to vary the PV P1H1:CurrentAO from -2 A relative to the original value to +2 A relative to the original value in five equal steps. The `execute` command specifies the SDDS output file and the pause time in seconds between measurements and variable changes. The output file contains column definitions for all measured or varied PVs and additional columns for the standard deviations, if requested in the measurement definitions. The data file consists of one data page with one row of data for each variable step.

The beam response to one corrector is obtained with a command as simple as: "**sddsslopes** P1H1.sdds P1H1.slopes -independentVariable=P1H1".

As an aside, **sddsexperiment** allows the variation of more than one variable at a time. Variables can be varied on a multi-dimensional grid by assigning increasing integer values for index_number, starting with 0, for each independent variable. This feature has been used in making a longitudinal phase space scan of the injection efficiency into the PAR.

Another feature of **sddsexperiment** is the shell execution of scripts in between measurements or changes to a PV. In general, these scripts prepare some accelerator system for measurement or change. This allows the design of a relatively complex experiment. For example, a script can be used to capture and process beam images for beam size measurements.

B. Generalized Feedback Control

A generalized feedback control was required to compensate the drift in the energy of the linac beam feeding into the PAR. For many months of running, the linac beam energy drift exceeded or was equal to the energy aperture of the PAR (± 1%). As a result, the injection efficiency into the PAR varied with time, and this made beam studies difficult. Since the energy drift was SLED cavity temperature related and therefore slow, the compensation was done with the program **sddscontrollaw** running on a workstation.

The first step of the general procedure is to identify the readback PVs and the control PVs related to the control problem. Then a linear response matrix of the readback PVs as a function of control PVs is measured, as described in the last section. The correction matrix required by **sddscontrollaw** is the inverse of this response matrix, and is computed by **sddspseudoinverse**, a generalized matrix inverter using the SVD method. The program **sddscontrollaw** is then applied with the correction matrix.

These steps are applied to the linac energy control. There are four BPMs in the beamline (LTP) transporting the linac beam to the PAR. Three of these are in non-zero dispersion locations (there is one bending magnet in the LTP following the first BPM). With all these BPMs, one can determine both the energy offset and the trajectory error, and correct for both. In controlling the linac beam energy there are many direct "knobs" available, such as pulse forming network voltage and rf drive. However, they have a slow response because they are motor driven. They are non-linear as well. Therefore we selected the SLED phase reversal timing variable for one of the linac sectors because this variable is roughly linear within its useful range, and its effect is prompt. In choosing the correctors for trajectory correction, we eliminate those that mimic the energy knob, such

as the LTP bending magnet, and any horizontal corrector close to it.

The following UNIX shell script was used to produce the correction matrix for the horizontal plane in the LTP:

```
#!/bin/csh
set correctorlist = \
    (LTP:H1 LTP:H2 LTP:H4 L5:SledTiming)
foreach corrector ($correctorlist)
    sddsexperiment $corrector.exp
    slopes $corrector.sdds $corrector.slopes \
        -independentVariable=$corrector \
        -excludeColumns=Time,$corrector
    end
sddscombine *.slopes LTP.R12.trans -merge
sddsconvert LTP.R12.trans \
    -rename=col,IndependentVariable=CorrectorNames
sddstranspose  LTP.R12.trans LTP.R12
sddsconvert LTP.R12 -rename=col,RowLabels=BPMNames
# calculate inverse
sddspseudoinverse  LTP.R12 LTP.InvR12 \
    -minimumSingularValue=0.01
sddsconvert LTP.InvR12 \
    -rename=col,RowLabels=CorrectorNames
exit
```

The response of the LTP BPMs to the correctors and the SLED phase reversal timing variable were measured with **sddsexperiment**. The files ending in .exp are user-supplied and are similar to the file P1H1.exp in the previous subsection. The data was processed by **sddsslopes** to produce files of slopes with respect to the actuator setpoints. The tool **sddscombine** combines these files to form a 4x4 matrix file. Renaming of columns (with **sddsconvert**) after some toolkit operations is necessary so that the user can keep track of the physical significance of the data. With the minimumSingularValue option in **sddspseudoinverse**, one can throw away singular values that are small relative to the largest one.

The command that executes **sddscontrollaw** is "**sddscontrollaw** LTP.InvR12 R12-feedback.out -timeInterval=6 -steps=30000 -gain=0.75 -controlQuantityDefinition=LTP:H.def - testValues=LTP:H.tests". R12-feedback.out is an SDDS file logging all PV values during the execution. The gain value is the fraction of correction to be applied at every step. The option control_quantity_definition requires a user-defined file which has cross references of column names to the actual PV names. This file is usually generated by an instruction in the **sddsexperiment** command file whence the correction matrix column names originate. To add robustness to **sddscontrollaw**, an input file describing tests for a list of PV values can be specified on the command-line to suspend **sddscontrollaw** temporarily. The file LTPH.tests contains tests of beam intensity and of rf power to the linac sections.

C. Orbit Correction

sddscontrollaw can be used for conventional orbit correction. In this case, the number of steps only needs to be a few. For PAR, the correction matrix was first derived from a measured beam response matrix rather than the model response matrix since, early in the commissioning period, we were uncertain of the tunes and of the calibration of the BPMs. By view-

ing the beam response data with **sddsplot** we were able to detect bad BPMs and correctors, which were then easily removed from the response matrix data file by referring to their names using SDDS utilities. The orbit in the PAR was corrected very quickly with hardly any beam time wasted on debugging the simple scripts and command files.

D. Other EPICS Applications

Mention should be made of other EPICS applications that have been written, not as tools, but as specialized programs to solve specific problems, such as the program controlling and correcting errors of the APS booster ramping power supplies [4]. These applications write out SDDS-protocol compliant data files, in order to take advantage of the SDDS toolkit.

V. ACKNOWLEDGEMENT

VI. REFERENCES

[1] M. Borland, "A Self-Describing File Protocol for Simulation Integration and Shared Postprocessors," these proceedings.
[2] L. R. Dalesio, M. R. Kramer, A. J. Kozubal, "EPICS Architecture," in *ICALEPS*, pp. 278–281, 1991.
[3] M. Borland, "Commissioning of the Argonne Positron Accumulator Ring," these proceedings.
[4] J. A. Carwardine, S. V. Milton, and D. G. McGhee, "Performance of the Ramping Power Supplies for the APS Booster Synchrotron," these proceedings.