GLAD: A GENERIC LATTICE DEBUGGER*

Martin J. Lee
Stanford Linear Accelerator Center, Stanford University, CA 94309

Abstract

Today, numerous simulation and analysis codes exist for the design, commission, and operation of accelerator beam lines. There is a need to develop a common user interface and database link to run these codes interactively. This paper will describe a proposed system, GLAD (Generic LAttice Debugger), to fulfill this need. Specifically, GLAD can be used to find errors in beam lines during commissioning, control beam parameters during operation, and design beam line optics and error correction systems for the next generation of linear accelerators and storage rings.

INTRODUCTION

I have been asked to present a paper on Model-Based methods and Artificial Intelligent (AI) methods for accelerator control. Being a teacher of T'ai Chi, a system of Chinese exercises based on the Yin and Yan principle, I am familiar with the Taoist saying:

Tao is Yin and Yan

It is natural for me to think of Model-Based methods and AI methods as one system of methods—the GLAD system. While thinking about the GLAD system, I made a list of the Yin and Yan pairs associated with accelerator control:

Yin	Yan
Beam line	Beam
Design	Control
High-level	Low-level
Commission	Operation
Play back	Real time
Beam	Parameter
Element	Strength
Look	Adjust
Off-line	On-line
Inverse-Modeling	Modeling
Interpretation	Analysis
Automatic	Manual
Solution	Problem
Rule-based	Trial-and-error
Prediction	Validation
Future	Present
Waste Prevention	Risk Reduction

The purpose of this paper is two-fold: (1) to describe the Model-Based control philosophy in terms of these Yin and Yan pairs, and (2) to propose the GLAD method as a practical way to upgrade any existing accelerator control system to become an intelligent Model-Based control system.

* Work supported by Department of Energy contract DE-AC03-76SF00515.

THE TAO OF MODEL-BASED CONTROL

The Tao of Model-Based Control is a generic way of controlling beam parameters using modeling and simulation codes interactively during commissioning and operation of a beam line.

Beam Line Design and Beam Control Codes

Every accelerator or storage ring system consists of a charged particle beam propagating through a beam line composed of bending, focussing, and accelerating elements. In the design stage, the effects due to errors in the beam line are simulated using modeling codes. For example, modeling codes are used to design an orbit correction system consisting of dipole correctors and beam position monitors (BPMs). During commissioning and operation, these same modeling codes can be used to find the errors in the beam line elements and to control beam parameters interactively.

High-level and Low-level Software

The software of a Model-Based control program can be divided into high-level and low-level software. High-level software is the modeling and simulation code for the design and control of an accelerator beam line. Low-level software is the application code for setting the strengths of the beam line elements and measuring the beam parameters.

Commissioning and Operation Goals

High-level software can be subdivided into two types: one for commissioning and the other for operation. The goals of commissioning and operation are not the same. The goal of commissioning is to find the causes of measured beam errors, while the goal of operation is to correct the error effects on the beam.

Here is one example. Often beam orbit errors are caused by magnet misalignment; and BPM reading errors. During commissioning, it is necessary to first use orbit simulation codes to find errors in the beam line elements (the sources). After these errors are found in the beam line elements, they can be incorporated directly into the "as-built" model. During operation, the same orbit simulation codes can be used to identify the best correctors and calculate the strengths needed to correct the errors. Since the success of the operation will depend on the accuracy of the as-built model, the primary objective of commissioning is to find an accurate model of the as-built beam line. [1]

Play-back and Real-time Applications

In general, the procedures to find the "as-built" model involve the following two-step procedure:

1. Measure specific beam parameters, and
2. Analyze the measured data.

The first step uses low-level real-time software and the second step uses high-level play-back software. The database of the control system provides an interface between high and low-level software.

A GENERIC LATTICE DEBUGGER (GLAD)

Today, numerous accelerator simulation and analysis codes exist that can be used to find the as-built model of a given beam line. To name a few, there is COMFORT,[2], DIMAD[3], MAD[4], PETROS[5], RESOLVE[6], TRACY[7] and TRANSPORT[8]. In general, the GLAD process will find the as-built model using the following procedure:

1. Measure orbits, tunes, profiles, etc.
2. Save the measurements in the database.
3. Translate data files to the input format of a particular code,
4. Analyze the measured data to find errors in the beam line or in the model.
5. Validate errors with beam tests.
6. Update the model.

In addition, if errors exist in the quadrupole strengths, the beam line can be "tuned" using the as-built model. The tuning process typically involves:

1. A calculation of the "lattice function" errors (such as mis-match in the beta-functions, eta-functions, etc.), and
2. An adjustment of the lattice functions to remove the error.

If quadrupole magnets are misaligned, they can be corrected using orbit correctors near the misaligned elements. This correction process typically involves the following steps:

1. Measure the orbit.
2. Identify and correct BPM offset errors.
3. Calculate the strength of the correctors using an orbit simulation code such as RESOLVE.

The GLAD process will be either an "off-line" process or an "on -line" process. When file translation (step 3 of the GLAD process described above) are done manually, it is considered an off-line process. When they are done automatically, it is considered an on-line process. The advantage of an on-line process is to reduce the turn-around time so that experimental validation can follow error prediction as quickly as possible.

The GLAD system can be used either manually now or automatically in the future. For manual applications of the GLAD system, the graphic interface allows a user to implement these procedures interactively. With "adaptor" codes to link the user interface to the database of the control system, the Model-Based commissioning and operation procedures can be implemented on-line directly. An example of manual application is to use RESOLVE to validate the model and COMFORT to tune the lattice. In the future, rule-based expert systems can be added to the GLAD system to perform these tasks automatically.

A Generic Interface[9] (GENI-X) is being developed to run modeling or simulation codes and to display the input and output data using X-Windows. To link a new module in the GLAD system to a given database requires two adaptor codes: DB-Get and DB-Put. The DB-Get adaptor code is used to translate files from the database format to the input format of the modeling code. The DB-Put adaptor is used to "download" the results into the database of the control system. In this way, the GLAD system can link up with any existing control system. In addition, GLAD can also be used for developing rules and procedures to find errors automatically, and to provide an accelerator system emulator for operator training.

Modeling and Inverse Modeling

There are two ways to use GLAD in the commissioning and operation of a beam line: Modeling and Inverse Modeling. In Modeling, GLAD computes the effect of the value of beam line parameters on the beam. In Inverse-Modeling, GLAD does exactly the opposite; it solves for the value of the beam line element parameters to best match the measured data. For example, during commissioning, Inverse Modeling is used to find the as-built model from the measured beam orbits (BPM data) and it is used to correct errors in the beam orbits during operation. Inverse modeling is also useful to restore beam parameters after a shut-down, hardware failure, or it is used to control the beam parameters in the presence of slow hardware drifts.

Analysis and Interpretation

For each data analysis application, the high-level software can be used to analyze beam data and to display the result graphically. Based on interpretation of the result, low-level software is used to implement the predicted adjustments. To validate the result, the beam parameters are remeasured. By comparing the measured data with the predicted result, the user decides what to do next. If the measured result does not match the prediction, the GLAD process (interactive look-analyze-interpret-adjust) continues until the as-built model is found and verified.

Manual and Automatic

Today a user must decide which beam parameter to measure, what procedure to use in the analysis, how to interpret the results, what to adjust, and when and where to iterate the GLAD process. In the future, AI methods can be used to perform these steps automatically to save valuable beam time. For example, automated error finding and beam control procedures can reduce the time it takes to recover the beam after a shut down or a hardware failure.

I am proposing the development of an AI tool kit called ASAP (Automatic System Analysis Program). Some of the AI tools to be included are Fuzzy Logic, Expert Systems, Neural Nets, and Genetic Algorithms. After the GLAD system is fully implemented, the user can commission or operate the accelerator system manually or automatically with the AI tools.

CURRENT GLAD APPLICATIONS

One typical demonstration of the GLAD process has been to find quadrupole magnet misalignment and BPM offset errors in the SLC damping rings and in SPEAR by analyzing orbit data with RESOLVE. For these cases, RESOLVE was linked to the SLC and the SPEAR control systems with adaptor codes. In this section, the use of RESOLVE to find magnet alignment and BPM offset errors in the SLC electron damping ring (NDR) is described. In particular, procedures and rules developed to analyze the beam orbits and to verify the predicted results are presented in this section.

Trial-and-Error and Rule-Based Procedures

Over the past fifteen years, special modeling and simulation codes have been developed at SLAC to make orbit corrections. All of these procedures work well. In particular, it is possible to use on-line Model-Based orbit correction procedures to reduce the residual closed orbit error in the NDR to less than a millimeter.

Unlike the orbit correction procedure, the beam injection procedure in the NDR is not model-based. An operator typically follows a trial-and-error procedure:

1. Kick the beam onto the axis of the ring by pulsing a kicker magnet,
2. Steer the beam manually along the ring axis using orbit correctors in the ring to establish a good first turn orbit,
3. Adjust correctors upstream of the kicker magnet to match the second turn orbit to the first turn orbit to obtain orbit closure.

In practice, both the closed orbit correction and beam injection procedures work. But there are two problems: first, the closed orbit resulting from optimizing the injection process is not the same as the closed orbit obtained from minimizing the RMS orbit error, and second, it is not possible to inject the beam onto the corrected beam orbit without beam loss. In either case, more than a dozen correctors are needed to steer the beam to the "good" injection orbit or to the "good" closed orbit (the NDR circumference is 35m).

To systematically investigate the beam injection/storage problem, a colleague (Jeff Corbett) and I measured several sets of first turn orbits with all of the horizontal correctors off. The orbit files were translated into RESOLVE format using adaptor codes. Using "Multi-Track" Analysis procedures [6], we soon found five BPMs with large offset errors and three misaligned quadrupole magnets in the NDR. In addition, we also identified three correctors that could be used to minimize the orbit errors caused by the misaligned magnets. Based on these model predictions, the following procedure was established to inject onto the horizontal machine axis:

1. Turn all borizontal correctors in the ring off.
2. Ignore the BPM readings with predicted offset errors.
3. Adjust the beam orbit and beam energy at the end of the transfer line to steer the beam onto the axis of the NDR, i.e. zero rms readings on the BPMs up to the first misaligned magnet. Look to see that the beam is deflected off-axis at the first misaligned quadrupole.
4. Once step 3 is true, adjust the first corrector to steer the beam orbit back onto the axis up to the second misaligned quadrupole. Look to see that the beam is deflected off-axis at the second misaligned quadrupole.
5. Once step 4 is true, adjust the second corrector to steer the beam orbit back onto the axis up to the third misaligned quadrupole. Look to see that the beam is deflected off-axis at the third misaligned quadrupole.
6. Once step 5 is true, adjust the third corrector to steer the beam back onto the axis. Look to see that the orbit in the second turn matches the orbit in the first turn.
7. Once step 6 is true, store beam and measure the closed orbit.

Prediction and Validation

The above procedure was given to the operators with a list of the three misaligned quadrupoles, the three chosen correctors, and the BPM offset errors. Following this procedure the operators were able to inject the beam on axis by obtaining the second turn orbit approximately equal to the first turn orbit within a few minutes. All of the steps went almost exactly as predicted. The operators thought our test was a success since it would take much more time and effort and many more correctors to accomplish the same result by trial-and-error.

Personal and Artificial Intelligence

The success of this experiment not only validated our predictions, it also confirmed the rules we developed for finding and verifying quadrupole alignment and BPM offset errors. As a result of our findings, two rules which will be used to automate the GLAD process are as follows:

1. A BPM is good if the predicted beam orbit agrees with the measured value for all tracks. A BPM is bad if the predicted beam orbit disagrees with the measured value for all tracks and the BPMs on both sides are good. If the difference between the measured value and the predicted value is the same for all tracks, then the difference is the BPM offset error.
2. If the beam is on axis (readings at the good BPMs are zero), the measured values at the bad BPMs are the offset errors.

In the case of the NDR, we found that the predicted offset errors at the bad BPMs agreed with the measured values. From this result, it is now possible to understand what the operators had to do without the knowledge of these offset errors. Since the operators normally used at least one corrector to steer the beam through the center of each bad BPM, at least five correctors had previously been used (incorrectly) to steer the beam. In addition, the operators used at least three correctors to compensate the three misaligned quadrupoles and two more to close the orbit. The total number of correctors therefore would add up to at least ten. With the knowledge of the alignment errors, the operator will be able to inject beam on axis by using only three correctors.

In addition, we noticed during the Model-Based beam injection/storage test that beam loss occurred at each turn in the extraction septum region. Since a localized loss

causes a beam centroid shift, it can be modelled as an abrupt "jump" in the beam orbit. Thus, in the presence of beam loss, the orbit of the injected beam can never be exactly equal to the closed orbit of a stored beam. These results suggest the following two "rule(s) of thumb" for injection/storage beam into a storage ring:

1. When the number of correctors are large, look for bad BPMs.
2. When the closed orbit can not be made equal to the injected orbit, look for beam loss during injection.

PRESENT AND FUTURE

Over the past few years, RESOLVE has been used to analyze beam trajectory data to find various types of errors in beam line elements and beam monitors at PEP, SLC, and SPEAR. In particular, procedures have been developed and tested to find dipole, quadrupole, sextupole, and rf cavity field strength errors, as well as displacement and rotational errors. Similarly, procedures have been developed to find BPM sensitivity and offset errors, and physical aperture restrictions. The development of these procedures is based on the knowledge of accelerator physics and ability to translate them into "if A then B" rules for data analysis.

Today, it is possible to compile these rules into an expert system to automate data analysis.[10] It is also possible to automate any rule-based procedure (such as the injection procedure described in the previous section) using an expert system.

In the future, other AI tools such as Fuzzy Logic, Neural Nets, and Genetic Algorithms can also be used for accelerator control. For example, Fuzzy Logic can be used for interpreting the result of the analysis since the rules may be more qualitative than quantitative (Fuzzy Rules). Neural Nets can be used to recognize errors or to handle exceptions. Adaptive feedback/correction systems can be developed using Neural Net models[11]. Finally, Genetic Algorithms can be used to train Neural Networks or to search for optimal solutions.

The development of GLAD, a Generic Lattice Debugger System, will allow us to analyze beam data with any modeling or simulation code available. In addition, GLAD will be the natural step toward developing rules and procedures for accelerator commissioning and operation, toward implementing AI methods, and toward developing automated rule-based procedures.

In conclusion, experience in accelerator control has validated a popular saying:

An ounce of (waste) prevention is worth more than a pound of (unnecessary) cure,

which also tells us that,

An ounce of knowledge of the cause of error is worth more than a pound of correction on the effect.

REFERENCES

[1] M. Lee, Y. Zambre, W. Corbett, "Accelerator Simulation Using Computers," SLAC-PUB-5701, August 1991.

[2] M.D. Woodley, "Control of Machine Functions or Transport Systems," in *IEEE Trans. NS-30:no. 4, Aug 1983* SLAC-PUB-3086.

[3] R. Servranckx et al., "Users Guide to the Program DIMAD," SLAC-PUB-0285, May 1986.

[4] H. Grote, F. Christoph, "The MAD Program (Methodical Accelerator Design) Version 8.4: User's Reference Manual," Iselin (CERN), CERN-SL-90-13-AP-Rev. 2, Aug 1991, 120 pp. Revised version.

[5] J. Kewisch, K.G. Steffan, "PETROS," DESY PET-76-09, DESY/Hamburg, 1976.

[6] Cf. M.J. Lee for information on RESOLVE.

[7] H. Nishimura, "TRACY: A Tool for Accelerator Design and Analysis," LBL-25236-MO, presented at European Particle Accelerator Conf., Rome, Italy, June 7-11, 1988.

[8] David C. Carey and Karl L. Brown, "New Features in Transport," SLAC-PUB-0091-Add., February 1983.

[9] S. Kleban, M.J. Lee and Y.B. Zambre, "GENI: A Graphical Environment for Model Based Control," SLAC-PUB-5126, Oct 1989. Presented at Accelerator Control: Int. Conf. on Accelerator and Large Experimental Physics Control Systems, Vancouver, Canada, Oct. 30-Nov. 3, 1989. In Nucl. Instrum. Math. A293:475-479, 1990; also in Vancouver Accel. 1989:475-479 (Journal).

[10] M.J. Lee, S. Kleban, SLAC-PUB-4539, Feb. 1988. Presented at 1st European Particle Accelerator Conf., Rome, Italy, June 7-11, 1988.

[11] D. Nguyen et al., "Accelerator and Feedback control Simulation Using Neural Networks," SLAC-PUB-5503, May 1991. Contributed to IEEE Particle Accelerator conf., San Francisco, CA, May 6-9, 1991.