LA-UR--87-3239

DE88 000485

TITLE: TOWARD AUTOMATED BEAM OPTICS CONTROL

AUTHOR(S): R. R. Silbar, T-5
D. E. Schultz, MP-1

SUBMITTED TO: Europhysics Conf.,
Villars sur Ollon, Switzerland
Sept. 28 - Oct. 2, 1987

TOWARD AUTOMATED BEAM OPTICS CONTROL

Richard R. Silbar and David E. Schultz
Los Alamos National Laboratory, Los Alamos, New Mexico 87545

Abstract

We have begun a program aiming toward automatic control of charged-particle beam optics using artificial intelligence programming techniques. In developing our prototype, we are working with LISP machines and the KEE expert system shell. Our first goal was to develop a "mouseable" representation of a typical beam line. This responds actively to changes entered from the mouse or keyboard, giving an updated display of the beam line itself, its optical properties, and the instrumentation and control devices as seen by the operator. We have incorporated TRANSPORT, written in Fortran but running as a callable procedure in the LISP environment, for simulation of the beam-line optics. This paper describes the experience gained in meeting our first goal and discusses plans to extend the work so that it is usable, in realtime, on an operating beam line.

1. Introduction

There has been much activity lately in applications of artificial intelligence (AI) techniques to knowledge-intensive domains [1-3]. We have recently begun to evaluate such techniques for solving problems in accelerator control. Because there have already been successful attempts using AI to control other complicated processes [4,5], one has good reason to hope for success in this venture.

AI programming techniques often contrast sharply with traditional computing approaches. One AI technique that we use is object-oriented programming (OOP) [6,7]. This encourages developing a computer model which closely resembles the real-world problem. Using OOP it is easier to debug, explain, and verify the model. It is also easier to add a new feature (object) to the model because a new object can inherit most of its characteristics and behavior from existing objects. The new object then needs only slot-value changes to reflect what is different about that particular device. Because of the partitioning of program behavior--the essence of OOP--different objects will not interfere with each other programmatically.

Another technique we use is symbolic modeling. Since such a model has causal relationships built in, actions leading up to an event can be easily described [8]. In contrast, the relationships in a numerical model are often structurally opaque. That is, intermediate steps in a calculation are not necessarily closely related to the underlying physical process [8], and this makes cause/effect explanations difficult. There are many problems, however, that have an elegant and efficient solution using traditional algorithms. For these problems one should not even consider AI techniques. Control of beam optics appears to be somewhere between the extremes of purely algorithmic and purely symbolic approaches. Thus, as in Ref. 9, the project described here uses a numerical model based on a Fortran code (operating in the LISP environment) to simulate beam line behavior. With the results of the numerical simulation always available to the inference engine, much of the decision-making in our model is symbolic in nature.

Using the Knowledge Engineering Environment (KEE) expert system shell, we have built a prototype knowledge base which describes the characteristics and the relationships of about 30 devices in a typical beam line. Each device is categorized generically, and pertinent attributes for each category are defined. Specific values representing static and dynamic characteristics for each device are assigned to slots in frames. These slot values are constrained by data type and any limitations or restrictions on the range of the data.

Relationships between the various beam-line devices are modeled using rules, active values, and object-oriented methods. Our knowledge base provides a framework for analyzing faults and offering suggestions to assist in tuning, based on information provided by the accelerator physicists (domain experts) responsible for designing and tuning the beam line. There is a general-purpose mechanism for determining device and beam-line status based on device-specific critical parameters. This approach simplifies knowledge acquisition by allowing the domain expert to concentrate on what is important about a particular device without getting bogged down in trying to specify rules for it.

A powerful graphical interface allows the operator to "mouse" on an icon for a particular item in the schematic of the beam line and obtain device-specific information and control over that device. The beam optics code TRANSPORT is used to model the beam line numerically, and any changes induced by the operator (or eventually, by the accelerator itself) are automatically updated on the operator's display.

Preliminary indications from using our knowledge base are that artificial intelligence techniques and traditional methods of numerical simulation can, and probably soon will, be successfully combined to provide a powerful tool for control of accelerators.

2. The Problem

The problem chosen [10] is the tuning of the first 30 devices in the LAMPF H^+ beam line, which transports protons from the Cockcroft-Walton ion source to the first emittance-measuring station. This relatively small beam line is of an appropriate size and complexity for a first prototype of an AI-based accelerator control system. The tuning goals are to minimize the emittance growth of the beam, to steer it, and to match the output emittance to the acceptance of the next section of the accelerator beam line. These constraints define a small region in the transport phase space which will provide an acceptable tune. Each time the ion source changes, the beam-transport parameters must be re-tuned to accommodate the slightly different characteristics of the new source.

To indicate the complexity of the problem under discussion, the major hardware in the H^+ beam line includes: two bending magnets, six steering magnets, eight quadrupole magnets, a beam deflector, an RF pre-buncher, four current monitors, an emittance measurement device, a beam-profile harp, two phosphor viewing screens, a beam scraper with four jaws, and an adjustable aperture. There are numerous other connecting and support devices, minor but vital to the correct operation of the beam line.

3. The Traditional Approach to Beam Tuning

Manual tuning of a beam line is an iterative process involving many steps: steering, adjusting quadrupoles, steering again, bringing deflector plates, jaws, and apertures to the edge of the beam, and then repeating the process. The data obtained from the diagnostic devices that measure beam characteristics are analyzed by Fortran programs running on the LAMPF VAX/VMS control system. The analyzed data are then used to generate beam envelopes and predict new tunes with a first-order optics code.

This information is available in a graphical or tabular form to the person tuning the beam line. Working with the correlations that can be identified in this data, along with knowledge of the desired "design tune" and use of particle-tracing codes, the operator seeks to find a solution that focuses and steers the beam from one end of the beam line to the other. This procedure works relatively well, but it is time-consuming and labor-intensive, requiring close attention by highly trained personnel. The operator must judge whether the successive iterations are converging to an acceptable solution. It is not uncommon to find that the converged-upon solution space is unacceptable; in such cases the work must be abandoned and the procedure started over. The "better" experts find many fewer unacceptable solutions, i.e., somehow know how to avoid the pitfalls (local minima in the parameter space). Perhaps some beam-line tuners are especially adept at extracting nuances from graphical data and exploiting them.

4. An AI Approach to Beam Tuning

We have chosen to use a hybrid of Model-Based Reasoning (MBR) coupled with the beam optics code TRANSPORT [11] to tackle the beam-line tuning problem. We represent the beam line using a symbolic model embedded in a KEE knowledge base. It describes the characteristics of and the relationships among the devices in the beam line. We categorize each device and define pertinent attributes for each category. Specific values are assigned in the knowledge base to represent each actual device. Relationships between devices are modeled using the techniques of rules, active values, and object-oriented methods. The knowledge base can be used to:

1) Simulate the devices in the beam line,
2) Identify faulty devices for repair,
3) Monitor progress in a complex tune procedure,
4) Advise on tuning actions ("What do I do now?"),
5) Explain advice given, and
6) Identify faulty devices as they affect the tuning procedure.

4.1 The Objects

Figure 1 shows some of the devices modeled, together with their class/sub-class relationships (denoted by a solid line). Dotted lines identify particular members of a class. For example, the class MAGNETS has sub-classes STEERING-MAGNETS and QUADRUPOLE-MAGNETS. Each type of magnet has members (specific instances), such as the particular device labelled TASM01H. All magnets have certain characteristics in common. These class-wide characteristics are inherited by the individual members of the class. The values of a member's slots reflect the state of that particular device.

4.2 The Active Values

Active values model some of the actions of the devices in the beam line. For example, there is a method, or procedure, associated with the active value attached to any slot (attribute) on any device that has a set-point and a tolerance. Active values ensure that, every time the value of the slot changes, the method is automatically invoked. The LISP code in this method could say, for example, that if the value about to be stored differs from the set-point value by more than the tolerance, then this device is a candidate for being the cause of any problem. Other evidence will be needed to narrow down the problem to a specific device.

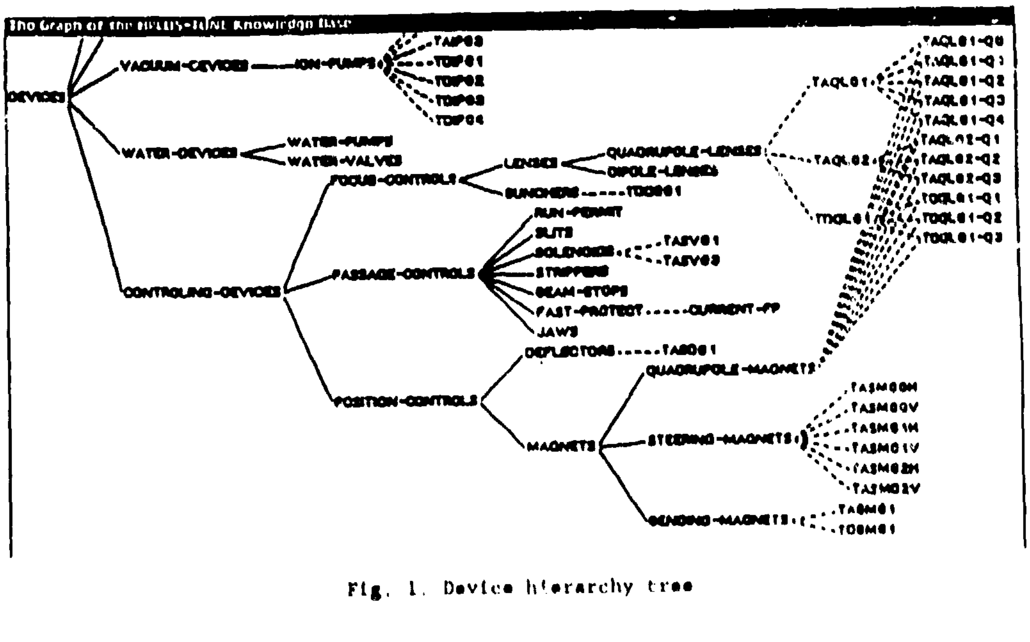

Fig. 1. Device hierarchy tree

4.3 The Reasoning

Rules capture heuristic knowledge. For example, the rule shown in Fig. 2 describes how to isolate a problem using knowledge about current monitors. The rule says that if you find a current monitor reading that is "not OK", then the possible causes of the problem are those devices that affect that current monitor (presumably, but not necessarily, upstream). Note the "wild-card" variables beginning with "?" in the rule. Even without knowing the syntax of the KEE rule system, it is relatively easy for a casual reader to determine what the rules mean.

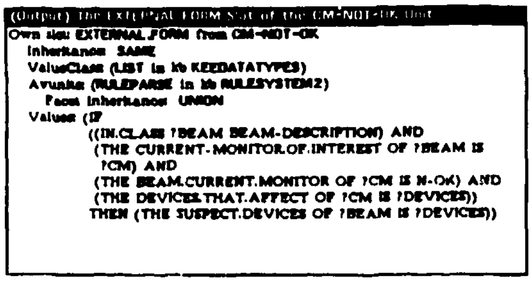

Fig. 2. A Sample Rule

We have developed general-purpose rules for two reasons. First, the specific detailed rules for some situations have not yet been determined. Second, general-purpose rules simplify knowledge acquisition. For example, domain experts are well aware of the critical parameters of each device but it is often difficult to extract from them specific rules about those devices. A device attribute that is a critical parameter is given facets reflecting the expected value, relationships, and resultant state value. This framework supports CHECK-FOR-GOOD and CHECK-FOR-BAD functions. CHECK-FOR-GOOD simply does an AND of all the critical parameters for a given device. Each critical parameter must match the good relationship between expected and actual values. Likewise, CHECK-FOR-BAD is an OR of all critical parameters. If any critical parameter has a value that falls into a bad relationship with the expected value, then that device is designated as having a bad status.

4.4 Simulating Real Data

Our knowledge base was originally developed on TI Explorer and Symbolics 3600-series LISP machines using the KEE development system. Recently it has also been ported to a microVAX AI workstation that has the ability to communicate with the LAMPF control computer. For the present, however, the knowledge base is not connected to the accelerator real-time values. For that reason, we designed a mouseable schematic to allow operators to select a device of interest and display a related image panel which displays the values of the interesting parameters for that device. The operator can then enter data for test purposes. He selects and changes a (simulated or, later, real) value by positioning the mouse cursor on its icon or image panel and clicking a button on the mouse.

4.5 Integration with Numerical Simulations

To tune the device (or to display changes in the tuning when a fault occurs) we make frequent use of the beam optics program TRANSPORT [11]. When a beam-device parameter changes, an active value associated with that parameter invokes a method that changes that value in a file representing the "TRANSPORT input deck". Another method then re-runs TRANSPORT, which writes its results to a standard output file. This file is then parsed by a third method, which extracts the new beam positions and envelopes, transfer matrices, and phase space ellipses. These are used by the active images of the knowledge base. In a nutshell, if the operator mouses on a beam element to change a parameter, the changes in the beam are automatically re-computed and re-displayed. (See Fig. 3 for an example.) The whole process takes only a few seconds; most of the CPU time involved is spent on setting up the FORTRAN process and accessing files, not on the actual TRANSPORT calculation.

Numeric outputs from the TRANSPORT program provide important pieces of knowledge needed to successfully select the proper course of action during a tuning procedure. For example, if TRANSPORT reports that the beam is off-axis at point B, we know that only devices upstream from that point can be the cause of the alignment error.

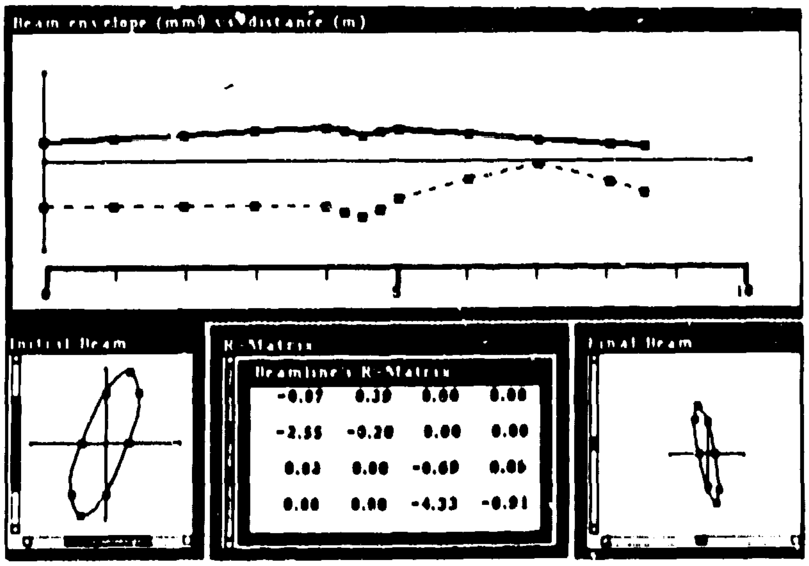

Fig. 3. Active display of beam envelope, etc., for a symmetric quadrupole triplet

5. Current Status and Future Work

The following components of our knowledge base have been built and tested:

1) A static model containing most of the information about the beam line and the characteristics of each device in it.
2) Image panels that allow simulation of test data.
3) A few rules using current monitor information to identify candidates for causing failure.
4) Active values that propagate device errors to affect the proper current monitor.
5) General rules to identify device status based on critical parameters.
6) Demonstration image panels that allow activation of the rule set and display of the conclusions reached.
7) Conversion of TRANSPORT to run in the LISP/KEE world.
8) Image panels displaying phase space ellipses and beam envelopes.
9) Mouseable schematic for operator interactions.

The symbolic model part of the prototype consists of approximately 400K bytes of code and was developed in about 3 man-months. An additional man-month was spent on converting and getting the 9300-line Fortran TRANSPORT code to run in the LISP environment. The interface between LISP and FORTRAN and graphics support routines take an additional 100K bytes. It is expected that the next stage in the integration of TRANSPORT with the knowledge base will take another 1-2 man months. While the parts of the knowledge base we have built so far handle only a very small fraction of the total problem, they nonetheless demonstrate the power of the integrated software development environment to create useful models quickly.

The next steps in our development of the knowledge base are to include:

1) Additional descriptions of the tuneup procedure.
2) Rules of thumb for choosing which step is most useful to do next.
3) Additional rules for determining device failure that use diagnostic tools besides the current monitors.
4) The identification and specification of the critical parameters for all appropriate devices.
5) Refinement coming from addition of more specific rules to handle less obvious and less frequent problems.
6) Completion of the connection to the control computer so we will be able to use live data from the beam line.
7) Other uses of TRANSPORT to confirm diagnosis.

6. Conclusions

The initial results of our work have shown that artificial intelligence techniques can be successfully combined with traditional methods of numerical simulation. While only a small part of the problem has been modeled so far, the interpretations given by the hybrid system match those of human operators. It is expected that additional rules will be able to capture more fully the expertise used by a beam-line physicist. Successful and efficient operation of future accelerators may well depend on the proper merging of symbolic reasoning and conventional numerical algorithms.

Acknowledgements

We thank Peter Clout for introducing us to this problem and for encouragement. This work has been supported by the U. S. Department of Energy.

References

[1] E. Rich, Artificial Intelligence (McGraw-Hill, New York, 1983)

[2] F. Hayes-Roth, D. Waterman, D. Lenat, et al., Building Expert Systems (Addison-Wesley Publishing Co. Inc., Reading, 1983)

[3] P. Harmon, and D. King, Expert Systems (John Wiley and Sons Inc, New York, 1985)

[4] H. Brand and C. Wong, "Application of Knowledge-Based Systems Technology to Triple Quadrupole Mass Spectrometer (TQMS)", AAAI '86 Proceedings, pp. 812-818

[5] P. Sachs, A. Paterson, and M. Turner, "ESCORT: an expert system for complex operations in real time", Expert Systems, Vol. 3, No. 1, 1986, pp. 22-29

[6] D. Bobrow, et al., "Common Loops--Merging Lisp and Object-Oriented Programming", OOPSLA '86 Proceedings, pp. 17-29

[7] D. Moon, "Object-Oriented Programming with Flavors", OOPSLA '86 Proceedings, pp. 1-8

[8] J. Brown, R. Burton, and J. deKleer, "Pedagogical, Natural Language, and Knowledge Engineering Techniques in SOPHIE I, II, and III", Intelligent Tutoring Systems (Academic Press, London, 1982), pp. 227-282

[9] S. Clearwater, "Developing a Hybrid Expert System for use at a Particle Accelerator Facility", Los Alamos National Laboratory Report LA-UR-86-1516

[10] D. Schultz, J. Hurd, S. Brown, "An Artificial Intelligence Approach to Accelerator Control Systems", Los Alamos National Laboratory Report LA-UR-87-739

[11] K. L. Brown, F. Rothacker, D. Carey, and Ch. Iselin, "TRANSPORT--A Computer Program for Designing Charged Particle Beam Transport Systems", Report SLAC-91, Rev. 2, UC-28 (1977)