STATUS OF MAPA (MODULAR ACCELERATOR PHYSICS ANALYSIS) AND THE TECH-X OBJECT-ORIENTED ACCELERATOR LIBRARY*

J.R. Cary,[a] S. Shasharina and D.L. Bruhwiler
Tech-X Corporation
Boulder, CO 80303, USA

Abstract

The MAPA code is a fully interactive accelerator modeling and design tool consisting of a GUI and two object-oriented C++ libraries: a general library suitable for treatment of any dynamical system, and an accelerator library including many element types plus an accelerator class. The accelerator library inherits directly from the TxDataSet library, which uses associative arrays to store defining parameters or strings and make them accessible by name. The GUI can access this data in a general way, allowing the user to invoke a window displaying all relevant parameters for a particular element type or for the accelerator class, with the option to change those parameters. A TxTransferMap object can advance an arbitrary number of dynamical variables through an arbitrary mapping. The accelerator class inherits this capability and overloads the relevant functions to advance the phase space variables of a charged particle through a string of elements. Among other things, the GUI makes phase space plots and finds fixed points of the map. We discuss the object hierarchy of the two libraries and use of the code.

1 INTRODUCTION

A user-friendly accelerator analysis and design tool would permit accelerator physicists to become involved in accelerator projects more rapidly. Tech-X Corporation has been developing such a tool under the auspices of the DOE Small Business Innovation Research Program. Currently, our application reads files in SIF format, permits modification of the description data through a Graphical User Interface, and subsequent writing of the data. The currently implemented analysis tools include geometrical survey, dispersion and Twiss parameter plots, and dynamic aperture analysis.

Mapa makes heavy use of utility classes in the Tech-X Standard library, libtxstd.a. This library has containers, strings, formulas, and linear algebra. Built upon this class library is libtxid.a, the Tech-X identifier and dynamical systems library. The TxID classes define intefaces for data holders and advancing arrays of real numbers through a dynamical map. Next comes libtxac.a, the Tech-X accelerator modeling library, which is built on top of the TxID library. Our accelerator modeling library has two basic classes, Element and Accelerator. In this library are also a number of analysis methods, which are implemented through a Visitor pattern.

These libraries are available under the terms of the GNU public license. They can be downloaded from www.techxhome.com. HTML documentation for our classes showing the hierarchy and describing our methods is at our website. We welcome input, suggestions for modification, and collaborations. These libraries make extensive use of many features of the C++ standard, including member templates and the Standard Template Library, so they compile on only those compilers supporting those features. At present these include the Kuck and Associates Compiler[1] and the egcs compiler.[2]

2 MAPA CAPABILITIES

Mapa presents a Graphical User Interface for accelerator modeling. This interface allows the user to read, set, and store data. The interface permits interactive plotting of the geometry, the Twiss parameters, the first and second order dispersions, and the dynamic aperture.

2.1 Reading and writing accelerator description data

MAPA reads data from and writes data to files in the Standard Interchange Format. It parses almost all of the features of MAD-8 files. This includes formulas, the standard descriptors (e.g., L, K1, TILT, etc.), and the inclusion in LINE hierarchies. The remaining feature to be installed is to parse “negative” elements, (reversed LINE’s, time-reversed magnets). It writes out this data to files also, but at present it writes out only the formula evaluations, not the formulas themselves.

2.2 Changing accelerator and element data

Accelerator and element data are changed through GUI interfaces. Below is shown the interface for changing the data of an element. At the left side of the window are the “available elements,” those that have been defined and had parameters set. Clicking the modify button brings up the interface on the right, a series of text boxes that permit users to change the values. Per usual GUI applications, the user need not remember the names of the data that he/she can change.

---

[a] Also, University of Colorado, Boulder, 80309-0390, USA.

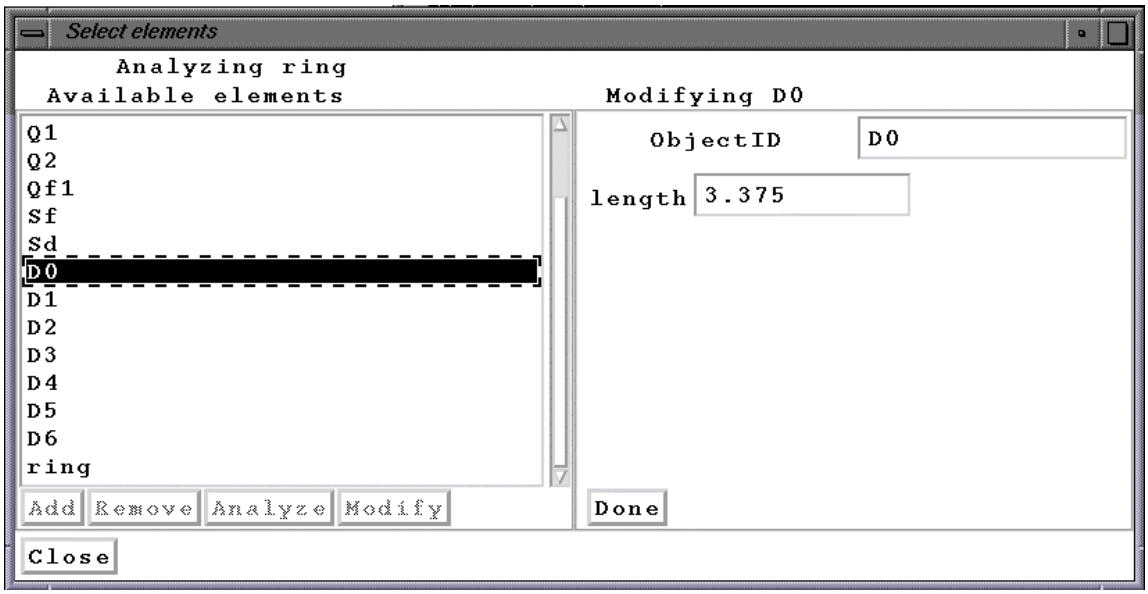

Figure 1: Element modification interface

At present elements have arbitrary length vectors of named integer, real, and string scalars. The GUI has been built to change this data structure. Our plans include adding input of formulas and named vectors of integer, real, and string scalars to more conveniently handle, e.g., multipole strengths. Because, as we describe later, all elements and the accelerator derive from standard data classes, the amount of code to write these interfaces is minimized.

2.3 Scalable analyses

Our "scalable" analyses calculate arrays of real valued data that are then plotted. These currently include the first- and second-order dispersion, the Twiss parameters, and the physical geometry. The data is plotted in its own window. The user is able to configure the plot interactively, zooming in or out to examine parts of the accelerator.

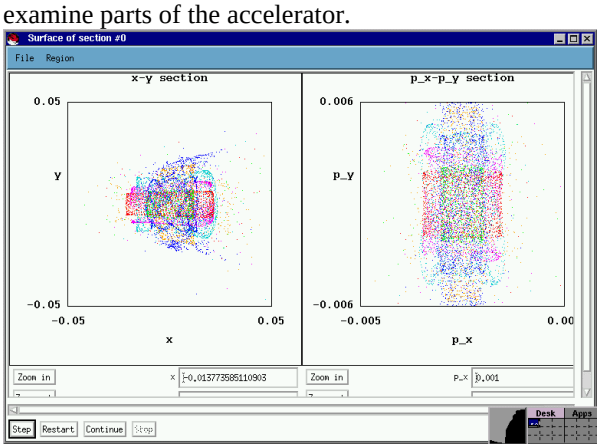

Figure 2: Surface of section showing dynamic aperture

2.4 Nonscalable analyses

We provide two "nonscalable" analysis, those for which the data is stored in pixel format. These analyses are both phase-space based. The first is a surface of section that gives a visual representation of the dynamic aperture. Secondly we have a "LifeTime" plot that shows chromatically the lifetime of various initial conditions in a circular accelerator.

3 TXSTD, THE TECH-X STANDARD LIBRARY

The TxSTD library (libtxstd.a) contains a number of utility classes, such as those for tensors, linear algebra, formulas, containers, and strings. The tensor and linear algebra libraries are templated over type and reference counted for flexibility and minimization of copying operations. Through template instantiation we provide libraries for both real and complex variables. More complex matrices, such as matrices over differential algebras, are trivially added.

Our TxFormula and TxFunction classes provide the implementation for parsing of formulas that may contain functions. The TxFunction class is extensible to add new functions of one variable. Currently the elementary functions are implemented.

We implemented containers and strings long ago, before wide availability of compilers that conformed to the current C++ standard and could handle the Standard Template Library. We are now slowly changing out these containers and string's as we now require compilers that implement the C++ standard.

4 TXID, THE IDENTIFIER AND DYNAMICAL SYSTEM LIBRARY

The library, txid.a, provides a base of classes for file I/O, data holding, dynamical systems and combinations of these items. The TxID class has the basics of identifiers and I/O. TxDataSet derives from TxID and adds data (int's, double's, and string's) that can be accessed by a name. TxMap contains the basics of a

dynamical map. TxTransferMap adds limits of validity, variable names, etc. The combination of TxMap and TxDataSet gives a TxDataAndMap, which simply combines the capabilities of these classes. Similarly, TxTransferMap and TxDataSet are combined in the class, TxDataTransMap.

Another part of this library is classes that hold data for plotting or output, typically array data. The base class TxPlotData has basic functionality. The derived class TxLinePlotData adds the ability to define lines that should be plotted. These are defined internally through TxDataSeries and TxLinePlotDesc objects.

5 TXAC, THE ACCELERATOR AND ELEMENT MODELING LIBRARY

The Element class inherits its geometry from a mixin class, ElemGeometry, and it inherits its data and dynamical systems interfaces from the TxDataAndMap class, which is part of libtxid.a. A Beamline is derived from the element class. The Accelerator class contains a single element, which may be a Beamline and so ultimately have many elements.

Our hierarchy is shown below. Derived from Element are ThinElements, for which the map is explicitly known, and Steppable's, through which the particles must be integrated. In either case the Advance method, for integrating particles through an element, must be present. Having this in Element simplifies our "Visitor" classes, which perform various analyses.

- ElemGeometry & TxDataAndMap
  - Element
    - Beamline
    - Drift
    - ThinElement
      - VerticalMonitor
      - VerticalKicker
      - ThnCavity
      - Marker
      - Multipole
      - Monitor
      - Kicker
      - HorizontalMonitor
      - HorizontalKicker
    - Steppable
      - TxQuad2
      - ThkSextupole
      - ThkOctupole
      - Quad
      - Bend
- ElementImg
  - SimpElemImg
  - BeamlineImg
- TxDispPlotVisitor
- TxSurveyVisitor
- TxTwissPlotVisitor

Our *Img classes contain the text images that can be used to define an element. These include variable names, numbers, and formulas.

Our *Visitor classes contain the analysis methods. The developer replaces the "MoreData" method of these classes to fill them with data. The GUI is then able to extract that data and plot it without knowing what it represents. This greatly simplifies addition of new GUI analyses to the MAPA application.

6 EXTENSIBILITY

We have achieved extensibility in MAPA through extensive use of the "Interface Pattern" and our TxCreator mechanism for creating a global registry of creatable elements. Thus, a user can define a new accelerator element and implement it in our application through a single line of code in the file TxacElements.C. Because all data is named and stored in name-associative arrays, the GUI code need be written only once, and then it is reused for all elements and even the accelerator itself. We use similar techniques for the various analysis classes. One can add to the plotting capability simply by creating a new derived class that fills in the data as requested by TxPlotData. Installation of this into the Mapa application involves a single line of code in the file TxPlotTrols.C.

7 FUTURE DIRECTIONS

With the basic structure implemented, we will continue to add features that can be easily derived from our existing structure. Depending on funding we intend to add design features to permit finding parameters to give certain desired operating values. We also intend to develop a more general class library interface that will allow insertion of other C++ class libraries (e.g., LEGO, BEAMLINE, MAD-9) into our GUI application.

8 SUMMARY AND CONCLUSION

Mapa is an extensible, object oriented accelerator modeling application with a graphical user interface. It has the basic modeling capabilities. It can be easily extended for additional types of studies.

The Tech-X accelerator class libraries have extensive capability in modeling, parsing, and data generation. They can easily be extended for additional types of elements and additional analyses.

REFERENCES

[1] www.kai.com.

[2] www.cygnus.com.