SIMPLIFIED CHARGED PARTICLE BEAM TRANSPORT MODELING USING COMMONLY AVAILABLE COMMERCIAL SOFTWARE*

D. Douglas#, K. Beard, J. Eldred, P. Evtushenko, A. Jenkins, W. Moore, L. Osborne, D. Sexton, C. Tennant, Jefferson Lab, Newport News, VA 23606, U.S.A.

Abstract

Particle beam modeling in accelerators has been the focus of considerable effort since the 1950s. Many generations of tools have resulted from this process, each leveraging both prior experience and increases in computer power. However, continuing innovation in accelerator technology results in systems that are not well described by existing tools, so the software development process is on-going. We discuss a novel response to this situation, which was encountered when Jefferson Lab began operation of its energy-recovering linacs. These machines were not readily described with legacy software; therefore a model was built using Microsoft Excel. This interactive simulation can query data from the accelerator, use it to compute machine parameters, analyze difference orbit data, and evaluate beam properties. It can also derive new accelerator tunings and rapidly evaluate the impact of changes in machine configuration. As it is spreadsheet-based, it can be easily user-modified in response to changing requirements. Examples for the JLab IR Upgrade FEL are presented.

DEVELOPMENT PARADIGMS

High-level accelerator applications have been generated within many organizational models, most of which comprise a "client/vendor" relationship. In this paradigm, a "client" (the user) contracts with a "vendor" (a software source) to provide a tool meeting defined requirements. The client states the requirements; the vendor is free to construct the application in any manner consistent with the defined criteria. Typically, the product will be general, robust, and secure. The user is, however, largely insulated from the development process, and often has limited ability to modify the tool. The overhead of changing or evolving requirements is therefore often high.

At the JLab FEL we have employed a different approach, wherein the emphasis is on development ease and speed, modifiability, and user accessibility. Applications developed under such a paradigm can be viewed as "just in time" products, created as "disposable", "near-run-time" codes intended for a specific purpose.

As an alternative to the client/vendor paradigm, consider then a "development" paradigm, wherein the user is provided a development *environment*, not an application. At or near run time, she develops an application tailored for use in a single instance. In this scenario, emphasis is placed on use of existing tools, ease and speed of development and modification, and user accessibility, rather than robustness, generality, and completeness, while maintaining required accuracy. Given the assumed ease of development under such a paradigm, the traditional requirements are not necessary to meet – if the software has a deficiency, it is modified at run time rather than off line!

In the context of accelerator operation and control, the "development paradigm" user is provided an environment and data access tools. With appropriate development methods, she captures needed data, construct tools for its analysis, and determines outcomes that are applied to the accelerator. Examples of such environments include the TCL/TK toolkit, packages such as MATLAB® and MATHCAD®, and comprehensive systems such as MATHEMATICA®.

Key to this approach is the realization that many high level operational applications use simple, readily captured data sets, involve only modestly complex processing, and produce results of limited scope and which do not require extensive implementation to realize within the accelerator. The core computations of many applications are simple, so it is possible to rapidly tailor applications to the case at hand, rather than attempt to generate applications meeting all possible combinations of events and exceptions.

We present below a common example of an accelerator application – a beam optics code. In its simplest form, such a beam-transport program is simply a tool to generate and manipulate tables of numbers; in the most general case, such codes are information systems describing accelerators. In all cases, the programs use (at least for linear optics) simple algorithms –computational complexity resides instead in user interfaces and data structures. Note, however, that the most desirable features of such programs – graphical input and output, data organization, and algorithms for data manipulation – are provided by commercially available spreadsheet programs. We have therefore constructed a machine model (for the Jefferson Lab IR FEL driver) using Microsoft Excel as the underlying computational engine.

EXAMPLE – AN EXCEL-BASED OPTICS APPLICATION

Microsoft Excel® is a familiar program with extensive analysis and development features. In fact, "… it is important to understand that Excel is not merely a spreadsheet. Excel is also a powerful object library that includes over one hundred advanced data analysis objects. With VBA [Visual Basic for Applications, an embedded dialect of Microsoft Visual Basic], developers can piece

*This work supported by the Commonwealth of Virginia, the Air Force Research Lab, and by DOE Contract DE-AC05-060R23177.
#douglas@jlab.org

together Excel's objects to create powerful information systems… There are currently thousands of Excel-based information systems in use ... throughout the world [1]."

Given that information system requirements of a large commercial organization can readily exceed those of a modest accelerator, we use Excel as a development tool for beam optics modeling. The following observations may be made.

- It is quite easy to generate rudimentary (linear) optics modeling capabilities within Excel, which is intuitive and robust, requiring little training overhead.
- Using Visual Basic for Applications (VBA), it is straightforward to extend Excel to incorporate any modeling capabilities provided using typical high level languages such as FORTRAN, C, or C++.
- Excel provides network, WWW, and database connectivity. Applications can therefore be made available to multiple users, with robust protection, security, and configuration control features.

The development cycle for such applications is quite short. The core model described here – which been in use for ten years – was generated in tens of hours. Additional features required from a few minutes (for emittance data analysis and including a simple skew-quad rotator) to a few hours (to implement an SRF cavity matrix model). The cost effectiveness of this approach is thus obvious.

FEATURES OF THE SAMPLE APPLICATION

Data Structures

Our FEL driver model was built using a standard Excel workbook. Within the workbook, individual worksheets are used to manage data tables describing a beam line. At present, the worksheets in use include:

- **elements** – contains a sequential list of the elements comprising the beam line, and definitions thereof. Data for an individual beamline element is entered in cells along a row assigned to that element, in a format similar to that used in legacy versions of TRANSPORT [2]. Control sliders are provided to allow interactive variation of element excitations. See Figure 1
- **matrices** – contains a sequential list of matrix elements for beam line elements. The expressions for matrix elements are coded in cells along a row corresponding to the element; element data is read from the **elements** worksheet.
- **products** – contains a cumulative product of the linear transfer matrix through the beam line.
- **betas** – contains a table of beam envelopes, dispersions, and phase advances; these data are presented graphically on the **elements** worksheet.
- **orbits** – contains a table of ray-trace data, giving the image of an initial propagated ray under the transformations in the products worksheet. Cells are assigned for input of BPM data from difference orbit measurements, and controls simulating the effect of correctors on the orbit are provided. The modeled orbit and measured data are graphically presented (Figure 2).
- **emittance** – contains a table propagating the beam matrix, fields to enter measured spot sizes, and fit functions to reduce the data when performing multi-monitor emittance measurements. Propagated and measured spot sizes are, as with other data, presented graphically (Figure 3)

Figure 1: Screenshot of **elements** worksheet showing lattice functions and sliders controlling quad fields.

The application also contains a VBA module for an Excel function to compute CEBAF SRF cavity matrix elements. Other functionality can be readily provided by the user. Additional VBA forms are used to define control buttons, sliders, and charts. The Excel "solver" serves as an optimizer; the user defines desired fit functions as the need arises and minimizes it by varying appropriate parameters. Matrix, beam, and lattice properties are thereby readily optimized, emittance data can be reduced, orbits corrected, and, with some ambition, multiple differential orbit measurements can be reduced to search for lattice imperfections. An artistic user can use the Excel "drawing" toolbar to sketch the beamline under consideration, as has been done in the example workbook.

CONCLUSIONS

We have presented an alternative to traditional high-level-application development paradigms. An example application developed under this alternative view was described. This application has been successfully used at the JLab FEL for the past ten years.

REFERENCES

[1] Wells, E. and S. Harshbarger, "Microsoft Excel 97 Developers Handbook," p. xxiii, Microsoft Press, Redmond, WA (1997).

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

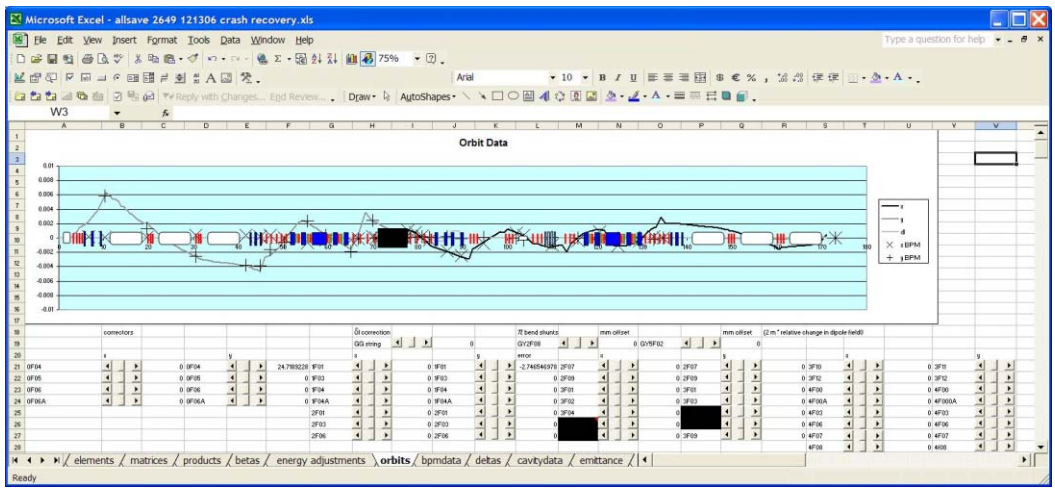

Figure 2: Measured and simulated difference orbit data from December, 2006.

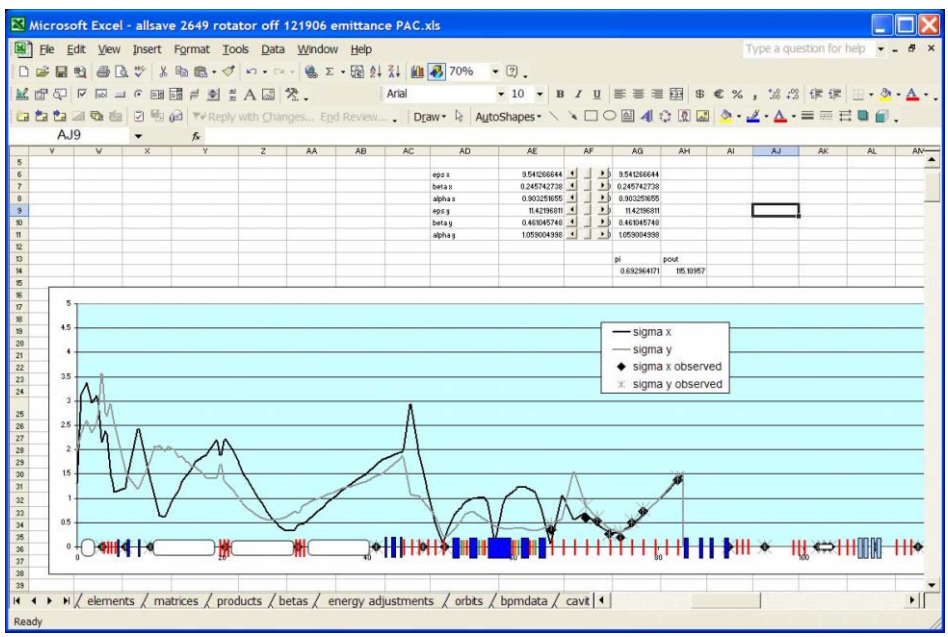

Figure 3: Emittance measurement from December 2006; initial beam envelopes and emittance are propagated and varied to fit measured data in a focusing-free region.

