

Online Modeling of the Fermilab Accelerators*

E. S. McCrory, O. Krivosheev, L. Michelotti, J-F. Ostiguy

Fermi National Accelerator Laboratory
PO Box 500, Batavia, IL 60510, USA

Abstract. Access through the Fermilab control system to beam physics models of the Fermilab accelerators has been implemented. The models run on Unix workstations, communicating with legacy VMS-based controls consoles via a relational database and TCP/IP. The client side (VMS) and the server side (Unix) are both implemented in object-oriented C++. The models allow scientists and operators in the control room to do beam physics calculations. Settings of real devices as input to the model is supported, and readings from beam diagnostics may be compared with model predictions.

Introduction

Beam physics models are accessed through the Fermilab control system in four ways: The Online Model (OLM), the Open-Access Model (OAM), standalone applications and the database of physics parameters. The OLM gives the user a means of performing beam physics computations on the Fermilab particle beam machines and directly comparing these calculations with real information from that machine. The OAM allows these same beam physics models to provide the readings for and accept the settings from controls application programs and is used for testing applications. A standalone application for fitting and tuning machine parameters has been developed for the Recycler Ring (RR) "Phase Trombone." The final way to access these models, while not strictly "online," has proven to be the most widely used access into our models: database tables of static beam physics. These tables contain all relevant beam physics values for a machine, for example, the design beam twiss functions.

The OLM and the OAM rely on a computation server to do the relevant beam physics calculations in a timely manner. This server is connected to the client via TCP/IP. In addition, the OLM uses the database.

Beam Physics Computation Environment

The online models rely on the *WXYZPTLK/Beamline* C++ class suite¹, developed at Fermilab. Briefly, *WXYZPTLK* provides a differential algebraic framework for performing particle propagation through a beamline to any order. Programmatically, higher-order calculations would behave like the first-order calculations that are relevant to operations. *Beamline* provides classes for representing beam line elements in a machine, and the various ways to combine these elements into hierarchical

structures. The object-oriented nature of this package has allowed significant flexibility in generating practical applications. The applications presented here are an example of this flexibility.

All accesses to our online models use the same machine description and the same calculation structure. Thus, questions asked of the model through one of the access points yields the same answer as from another access point. Furthermore, if more details are needed, a special-purpose offline application can be quickly constructed from the same classes.

Calculations are benchmarked against MAD². A parser has been written to translate lattices specified in the MAD input language to C++.

The computation models run on dedicated, high speed Unix workstations. There is a primary server workstation used for regular operational request and several backup servers for debugging.

Overall System Architecture

Since the Fermilab ACNET control system is based on legacy VAX/VMS computer, this environment is not practical for computationally intensive beam physics

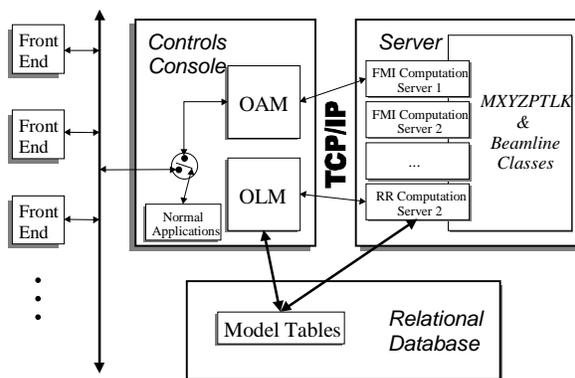


Figure 1, The Online Model Architecture

calculations. A client/server scheme has been developed to perform the required computations on Unix servers, while leaving the user and controls interfaces on the VAX/VMS X-Windows consoles—an environment familiar to control room personnel. This network architecture is shown in Figure 1. Communications between VMS and Unix is done via an ASCII-based TCP/IP network protocol. The components are:

1. The computational model on the server, which performs the beam physics calculations requested directly (for the OLM) or implied indirectly (by the OAM). The parts of this structure are the "server process" for getting the proper computational task going, the "computational

* Work supported by the US Department of Energy, contract # DE-AC02-76CH0-3000.

tasks” for performing the desired computations on the specified machine, and the underling *MXYZPTLK/Beamline* library.

2. The client side OLM and OAM take advantage of the OO framework. These are written in C++ under VMS.

(a) OLM: Specific application programs provide the user interface for launching the calculations and viewing the results. We have six such applications, written in C++ under VMS. Writing a new application involves the extension of a two of the ~90 classes in the OLM library. The applications can also perform the appropriate control system access of the readings (beam diagnostics devices) and the settings (magnets) of real devices.

(b) OAM: The Fermilab control system has a service that allows redirection of the information that a user application sends and receives to a software model. Our OAM uses this service. Writing a new OAM is more complex, see below.

3. The static database of beam physics parameters is kept on a relational database that is globally accessible. This database is accessed by individual users and by the OLM.

4. The network communication between the client and server is an ASCII-based protocol on top of TCP/IP.

Online Database of Physics Parameters

The most widely used access to beam physics computations has been the static databases of machine parameters. Calculating the data for these tables is an important aspect of debugging the beam physics model for a new machine. Thus, these tables come very early in the development of an OLM. C-callable functions have been created for application programmers to access these tables without direct knowledge of SQL. Many applications for the machines at Fermilab rely on these tables.

The sorts of data that are present in these database tables are the physical names, positions and alignment of the beam line elements in the lattice and the beam physics, like the orbit, twiss functions, R-matrix and covariance matrix. These physics parameters are available at every element in the lattice of each machine.

Online Models

Independent, abstract and generic class hierarchies have been developed in C++ on both the client and the server. The client side deals with the user interface, the connection to the control system, the communications with the server and the communications with the database. The server side deals with responding quickly to calculation requests by the client, communications with the client and communications with the database. The bulk of the data transmission between the client and the server is done through a relational database, with commands passed via TCP/IP over a simple, ASCII-based protocol.

The VMS Client

Features of the VMS client applications include³:

- An independent model for each user;
- Display of beam physics parameters, both textually and graphically;
- Email of any text to the user;
- Comparison of orbits between the model predictions and BPM readings (see Figure 2, an example taken from the FMI 8 GeV injection line);
- Selection of the device types the user wants to see;

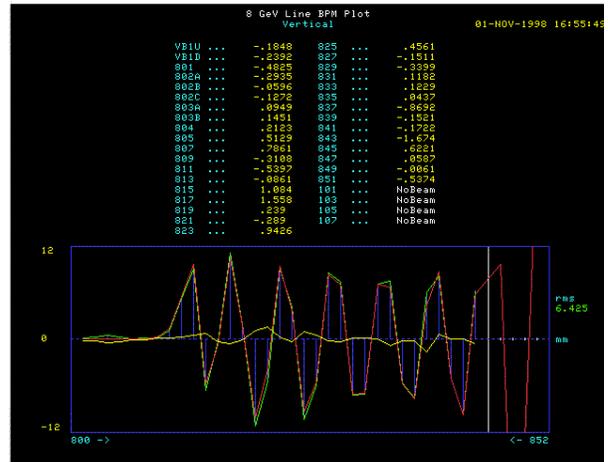


Figure 2, The Online Comparison between the model of the MI 8 GeV line and the BPM readings for it.

- Selectable calculations for the machine as a “ring” or as a beam line;
- Configurable “launch” parameters for the beam.

The Unix Server

The model server tasks run on relatively high-speed Unix workstations. These tasks respond to requests from the clients on a specific port number on the server. The first request tells the server which particle beam machine the client wants. This server task launches the model server application suited for this specific machine. This model server task communicates directly with the client, performing the desired calculations, changing the specified model devices in the way the client has requested, etc. The client sends these requests to the model server task via an ASCII-based protocol on top of TCP/IP.

The Role of the OLM Database

There often is a lot of data that would need to be transferred to the client when the server is done with a calculation. We start with a copy of the default, unaffected database tables. As the client changes the devices in the machine model, the server changes the values in the appropriate database tables. Then the client may use SQL queries to select the values of interest, e.g., the position of the beam at the BPMs.

Open Access Models

A feature of the Fermilab ACNET control system is the ability to change a user console's connection from the real hardware to an internal model. This type of model-based redirection is referred to as an Open Access Model (OAM). At least two desirable results are accomplished through an OAM: to test algorithms in online controls applications and to provide a meaningful way to train operators and scientists. OAMs have been written in C++ for the Fermilab Main Injector (FMI) and the RR.

VMS Side: The OAM

The emulation of the control system and of all the devices in a machine is handled in the OAM⁴. An OAM performs essentially two actions:

1. To accept settings for devices and interpret these settings into changes in lattice values in the model, and
2. To provide model-based beam sensor readings.

TBT and Tunes for the FMI

An example may clarify this structure. An ongoing OAM project is to mimic the behavior of the FMI for measuring the tune of that machine. The tune measurement program can thereby be tested. This program will configure a kicker to deflect the circulating beam for one turn. Then, the position at a BPM is measured over subsequent turns. A Fast Fourier Transform is applied to these data to determine the tune of the machine.

The OAM intercepts the network message(s) that would cause the pinger to fire, and the model server is informed. The request for the turn-by-turn data from the BPMs is also intercepted, which is interpreted as a request for a certain number of turns of turn-by-turn data. The model server performs this calculation and sends the data back to the OAM. The OAM takes these results and packages them to look like a network reply from the BPM controller. The application receives these packets and is fooled to think that these data came from the real world.

This is a good test for the OAM because the FMI BPMs have some well-understood mixing behavior. The OAM accurately models this effect. Preliminary tests of the FMI Tune application show that the effect of these BPM irregularities is small.

Fitting and Tuning With Standalone App

A different access into our modeling has been created for the RR, see Figure 3. The RR is based on permanent magnet technology. The tune is changed by adjusting the strength of five electromagnetic quad families in a special insertion called the "phase trombone." A fitting scheme has been implemented to enable this calculation. Emphasis has been placed on allowing the user to guide this inherently nonlinear fitting process in a convenient way. The VMS/Unix client/server has been bypassed: the application runs directly on the Unix server and the display appears on the controls console.

This program is a specialization of a general-purpose fitting and tuning program. It has been constructed to facilitate the calculation at hand. The general program reads the same lattice specifications and uses the same

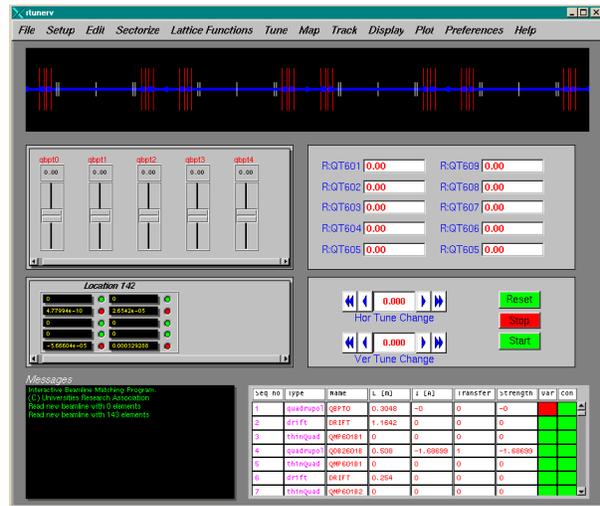


Figure 3, The Recycler Ring (RR) Phase Trombone Tuning Application

calculation engine as the online models.

Observations and Conclusions

OLMs exist at Fermilab for the following machines: Main Injector 8 GeV Line, FMI, RR, Tevatron fixed target mode, Tevatron collider mode and the Antiproton Source Accumulator. New OLMs have been added quickly. Open-access models exist for the FMI and the RR.

The OAM has not been applied to more of our systems because the level at which redirection occurs is entirely too close to the hardware. In order to make the OAM work, it must be known in great detail how a front end replies to a specific type of data request.

Our long-term goal is to provide a consistent environment to construct models and to do beam physics calculations. Since the Fermilab control system is evolving and will be changing considerably in the near future, we will be tracking those changes. We expect our architecture will provide the flexibility necessary to follow this transition.

References

- ¹ "MXYZPTLK version 3.1 User's Guide: A C++ Library for Differential Algebra," By Leo Michelotti, Fermilab publications #FN-353-REV, on web as <http://fnalpubs.fnal.gov/archive/1995/fn/FN-353R.html>.
- ² MAD: Methodical Accelerator Design program, by C. Iselin H. Grote at CERN, http://wwwslap.cern.ch/~fci/mad/mad_home.html.
- ³ The client class hierarchy is documented fully on our web site at <http://www-ap.fnal.gov/model>.
- ⁴ The user may turn on or off the redirection at any time. An application under redirection is marked with a prominent yellow line through the main screen.