AUTOMATIC BEAMLINE CORRECTION

T. Kobilarcik, J. DeVoy and C. Moore
FNAL, Batavia, IL 60510, USA*

Abstract

In the MiniBooNE experiment, a simple linear algorithm is used to maintain the correct beam position throughout the beamline and at the target. The algorith is explained, and key features of the implementation are discussed.

INTRODUCTION

Experience has shown that in any fixed target experiment at Fermilab, the primary beam tends to "drift" over time. There are many causes for this drift, such as diurnal variation in temperature affecting control cards. Traditionally, it was the role of the operations group to monitor the beam position and make apropriate corrections.

When several HEP programs are run simultaneously, the number of operators needed to monitor the beamlines becomes large. Additionally, the work is extremely tedious—watching several monitors, making small corrections (and usually only being able to correct one position at a time). Even when features such as "three bumps" are incorpoated in the control system, an operator can only adjust one position at a time.

Thus, making corrections to a beamline is a job well suited to automation. The response of the beam to changing individual magnets can be well characterized; the beam can be continuously monitored; appropriate limits can be placed so that the program "calls an expert" (for example, by setting an alarm) if conditions change too severely.

THE ALGORITHM

The algorithm used is simply the inversion of a set of linear equations.

A set of magnets, b, and a set of beam positon monitors, x, are chosen. As each magnet is varied, the change in beam position at each beam position monitor is recorded. This leads to the linear expression:

$$\delta x = M \delta b \tag{1}$$

To implement the algorithm, one simply inverts the equation. Note that cross-plane coupling is naturally accomodated in this algorithm.

Exact Solution versus Least Squares Fit

One will note that in order to implement the algorithm, the matrix, M, must be invertable. A commonly asked question is "Why not implement a least-squares fit?" There are several reasons to choose an exact solution over a least-squares fit:

- An exact solution is just that—it always puts the beam where it should be, not just close.
- Having more beam position monitors than adjustable magnets provides no additional usable information.
- Having more adjustable magnets than beam positon monitors implies that at some point along the beamline the trajectory is not known.
- The algorithm will fail if any one BPM fails. This is good—when instrumentation breaks it must be fixed.

Reasonable people may disagree with the relative merits of an exact solution versus least squares fit. Ultimately, one must make a decision and evaluate the outcome. Experience with the exact solution algorithm at Fermilab has shown that it is highly reliable [1], and thus was chosen for MiniBooNE.

General Implemenation Concerns

Although inverting the set of equations gives an exact solution, the measured changes in beam position are never exact. Thus, one should never implement the full calculated change every spill. The present correction program also allows the user to specify a tolerance and a convergance factor for each beam position.

Thus, the corrections will not be applied unless the beam position is out of tollerance at one or more locations. The tollerance is usually 2 to 3 times the RMS position change during stable running, although at tight apertures the tollerance is smaller. Furthermore, only a fraction of the correction (typically 80%) is made.

Additionally, one must check that beam is actually present when the measurement is made, and that no magnet will beset beyond its operating limit.

IMPLEMENTATION SPECIFIC TO MINIBOONE

The generic name of the program is "Autotune". Variations of this program will be implemented for various beamlines as needed; the MiniBooNE autotune is the first. Autotune written in Java using a client-server model. The server part runs inside an Apache Tomcat servlet engine. The client runs on a user's desktop and communicates with the server via Xml-Rpc.

The Autotune Server

The Autotune server runs continuously inside an Apache Tomcat servlet engine. In simplest terms, it is an infinite loop that monitors the position of each beam pulse and, if

* Operated by Universities Research Association, Inc. under contract number DE-AC02-76CH03000 with United States Department of Energy.

necessary, computes and makes new settings for the trim magnets.

For the MiniBooNE beam, a complication is the fact that the beam is composed of a series of "pulse trains". Each pulse train contains a set of pulses arriving at 15 Hz intervals, with the trains separated by 2-3 seconds. Trying to correct each individual pulse at that rate is not feasible. Consequently, the pulses in each train are averaged and the correction applied before the next pulse train arrives.

The average of the pulses is a weighted average of the positions of each pulse in the train. The weight for each pulse is determined by a combination of the intensity and position of that pulse. Three weighting algorithms are currently defined:

- Use the intensities only. All pulses below a threshold intensity are given a weight of zero; those above are given a weight of one. Note that if all the weights are zero, we effectively assume that no beam is present.
- Compute an initial set of weights and the average position using the above algorithm. Any pulses farther from this average than a threshold distance are reassigned a weight of zero, and the average position recalculated.
- Like the above, but the threshold distance is based on a multiple of the rms of the initial average, rather than being a simple constant.

The parameters used for monitoring and controlling the beam (averaging algorithms, cut thresholds, lists of BPMs and trims to monitor, the matrix, etc.) are stored in a set of "control files" (implemented as a set of relational database tables). Any number of control files may be kept. The operators may load, unload, and modify the files as necessary.

The control loop operates as follows:

1. Initialize. Load the control file designated as "active", and initiate data acquisition.
2. Wait for a pulse train to arrive.
3. Get the position and intensity of each pulse as returned by each position/intensity monitor.
4. If any returned an error code, goto step 2. Else calculate the average position of the beam at each monitor.
5. If the average position of each pulse is within tolerance at each monitor, then no tuning is ncesssary. Goto step 2.
6. If the average position of any pulse is farther from the nominal than a given limit then we will not move the beam. Goto step 2. Rationale: if the beam is too far from the desired position at any point, it is assumed that the operating conditions have changed, or that something is seriously amiss. In either case operator intervention is indicated.
7. We will attempt to move the beam. Compute the change in trim magnet settings using the algorithm described in section 2. Note that the deltas in the settings are multiplied by the convergence factor (setion 2.2). The convergence factor is currently a constant; in the future it may be made a function of the distance of the beam from the nominal. This would effectively merge steps 6 and 7.
8. Get the current setting of each magnet and add the delta computed above.
9. Set the new current for each trim magnet. Note that the operators have the option to suppress this step. This allows them to confirm that the calculations are correct before taking the program "live".
10. Goto step 2.

Testing indicates that the above loop takes about 200-300 ms.

The Autotune Client

The autotune client runs on a user's desktop and graphically displays the progress of the above loop. The horizontal and vertical position of the beam between adjacent monitors is indicated by a color-coded line: Green if the position is within tolerance at both monitors, Yellow if the intensity is low (at either monitor), Red if the position is out of tolerance (at either monitor).

Control buttons and dialog boxes are provided to allow the user to:

- Tell the server to suppress the setting of the trims (step 9) in the above loop.
- View the current readings (and possibly settings) in tabular format. This provides a more analytic view than the graphical display on the main screen.
- Create, modify, delete and load control files.
- Designate certain individuals (e.g., operators) as having the authority to make changes to the control files.
- View a log file of beam settings. That is, the server will keep a log of the times it has moved the beam. This gives an operator the opportunity to restore the trims to a prior state.

CONCLUSION

A simple algorithm has been implemented to correct for beam motion in the MiniBooNE experiment. The algorithm is easily extended to other beamlines.

REFERENCES

[1] T. Kobilarcik, "Preliminary results of Stability Study for the KTeV Beam", FERMILAB-TM-2037