Orbit Correction Technical Report

Robin Newhouse[*]
Department of Computer Science, University of British Columbia
6224 Agricultural Road, Vancouver, British Columbia, Canada, V6T 1Z1

(Dated: February 26, 2015)

This report summarizes the process in which additions were made to Open XAL's Orbit Correction application for the TRIUMF laboratory. TRIUMF's beam physics group requisitioned an Orbit Correction application and provided a list of requirements that it should meet. After investigation into Open XAL's existing orbit correction application it was decided that almost all of these requirements were met, needing only a few modifications. The major modification made was introducing the option of applying weights to the responders used in the optimization algorithm. The implementation was successful based on rudimentary testing.

[*] robinnewhouse@gmail.com

I. INTRODUCTION

In accelerator physics, beam steering (i.e. orbit correction) is a fundamental task. Without proper direction of the particle beam throughout the beamline, very little else could be achieved. This paper contains a brief technical report describing the process used to improve upon an existing beam steering application. At the time of this writing, Canada's TRIUMF laboratory is in the process of constructing and commissioning the Advance Rare IsotopE Laboratory (ARIEL) and its associated linear electron accelerator (e-linac). To aid commissioning and operations it was decided that this project would make use of Open XAL (footnote the project page). This enables the development and use of high-level accelerator applications to interface with the lower-level Experimental Physics and Industrial Control System (EPICS).

The beam physics group at TRIUMF, responsible for commissioning the linear accelerator, requested that an orbit correction application be developed with this high-level application platform. After some investigation it was decided that the existing orbit correction application met the desired requirements and that the most efficient use of resources was to understand and modify this application.

This paper describes the modifications that were made and the parts of code necessary to contextualize them. In particular the problem and implementation of weighted responders is discussed. This is by no means a complete description of the application's behavior and capabilities. Please consult the Orbit Correction user manual for more information [3]. Section ??? describes the beam steering problem and some approaches to it. Section ??? discusses the collection requirements and the decision of which modifications would be made. Sections ??? and ??? discuss the implementation and testing of these modifications and give descriptions code relevant to the context.

II. BEAM STEERING PROBLEM

The problem of beam steering is perhaps one of the most important in accelerator control. The primary objective is to minimize deviations of the beam from the center of the beam pipe, that is, to minimize the root mean squared (rms) of the orbit [6].

The system can be described by a collection of actuators (e.g. beam steerers) and responders (e.g. beam position monitors). A response matrix can be defined to describe the incremental action of an actuator on a set of responders. If we assume the response is linear, we can define the system by Ax=b, where x is the vector of actuators, b is the vector of responders, and A is the response matrix. Note that we assume the problem is overdetermined, that is, there are more responders than actuators (a safe assumption in this setup).

The primary objective of a steering algorithm is to find a solution to this equation that minimizes some parameter. In many situations, one would minimize the rms, which involves minimizing the two-norm of the residual. Formally we find x such that $||b - Ax||_2$ is minimized [5]. This naturally leads to a number of linear algebra-based solutions including singular value decompositions, eigenvalue decompositions, Householder transformations, etc. in order to find the optimal solution. Unfortunately, using linear algebra techniques restricts the addition of non-linear constraints that naturally exist in steerers.

In the implementation used in Open XAL, a combination of linear programming algorithms is used to search for the rms minimum while simultaneously (paying attention to) the constraints. In private correspondence with the application's author, the decision to not use a linear algebra approach was explained. First, though it is possible to capture the nonlinear constraints (steerer limits) in a linear problem, it is prohibitively difficult. Also, in the design of the application, it was explicitly intended that additional nonlinear objectives might be introduced in the future.

III. REQUIREMENTS COLLECTION

At the beginning of this project, a requirements document was produced by the customer (Chao) outlining what features should exist in the final product, some of which were seen as lower priority. These requirements were discussed until I felt I had a reasonable understanding, as a developer, of the high-priority requirements. Before I began designing an application which captured these requirements, I first investigated the existing orbit correction application packaged with Open XAL. The following paragraphs discuss the process leading to the decision to use the existing application and introduce minor modifications.

The following summarizes the requirements extracted from the initial requirements document and subsequent discussions ("optional" requirements omitted).

1. Given a set of actuators, responders, and desired orbit, the program should find the optimal solution to match the orbit
2. Solution should be based on a singular value decomposition of the response matrix
3. User should be able to enable/disable particular actuators and respondesr
4. User should be able to apply a linearly scaled fraction of the solution
5. Correction target may be not simply zeros, but instead a predefined orbit pattern
6. Should be able to view predicted results before application
7. Verify corrector range is not exceeded before application of values
8. Include an "undo" button
9. Save rceord to file for offline analysis

After exploration of the Open XAL Orbit Correction application, it was determined that most of these requirements were met. In many places there were useful features that had not yet been considered by TRIUMF. Some notable features not present were: a linear algebra-based algorithm was not used (see section (beam steering problem)), weighting of responders was not an option, there was no "undo" button, and no option to save a record of the steering solution.

To decide which feature would be addressed it was important to account for complexity, priority, and available time. Weighing these options and discussing with the various stakeholders, the beam physics group decided that introducing the weighted responders was the most appropriate task. Development of an entirely new algorithm could become too complex with little advantage over the existing solver. An undo button and a save button would not require enough understanding of the program to fulfill academic requirements (footnote: this project was done in the context of a directed studies computer science course). The rest of the requirements were met to a reasonable extent and it would be redundant to implement them again.

The use of weighted responders is useful when correcting an orbit by rms minimization. In this case Beam Position Monitors (BPM's) are used. Some sections of the orbit are more sensitive to deviations than others, and it may be desirable to capture this in the problem description. For example, if the exiting angle of the beam is significantly more important than deviations in the transport sections, one could weight the final two responders heavily.

IV. BASIC CODE STRUCTURE

This section will summarize the relevant aspects of the code at a very basic level. In particular, important data structures and sequences will be presented. By necessity, what follows may contain domain-specific language. For context, please refer to the Orbit Correction user manual (in progress). Initially, when an accelerator sequence is selected, information about the sequence is contained in an `OrbitModel` object, which itself contains lists of elements designated as responders (`BpmAgent`) and actuators (`CorrectorAgent`). `OrbitModel` also contains a `Flattener` object, which is the class responsible for executing an orbit flattening. The "flatten" method in `Flattener` contains the sequence with which the program performs an orbit correction. This is the first place one should look when attempting to understand the program. To correct an orbit, the program uses Open XAL's generic solver algorithm, `xal.extension.solver.Solver`. This has been used successfully for over a decade and has proven to be robust and effective [4]. To understand the `Solver` class, an important data structure to be aware of is the `OrbitObjective` class, which extends `xal.extension.solver.Objective`. This object is used to specify the desired values to be minimized by the `Solver`. The objectives used in Orbit Correction are the x and y rms displacements. There is an additional `CorrectorDutyObjective` that attempts to minimize the magnitude of the load on all correctors. As well as the `OrbitObjective` objects, the `Solver` is passed a collection of `Variable` objects to optimize, which in this case are the collection of `CorrectorAgent` objects. These `CorrectorAgent` objects contain the limits of the correctors and the solver will only search for a solution within those limits. The `Solver` then uses an iterative technique consisting of different algorithms used to search for the global minimum of this problem. Some examples of the algorithms used are: a Nedler-Mead Simplex method [2], an "acceleration-step" method by Forsythe and Motzkin [1], and a randomized search algorithm with a shrinking search window written by Tom Pelaia. The method used in the next iteration is selected randomly, weighted by previous success.

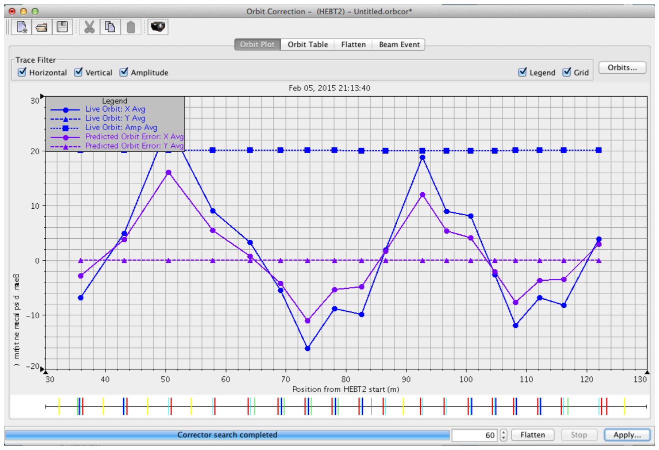
V. IMPLEMENTATION AND TESTING

Various options were considered when deciding how to implement weighted responders. When deciding upon a final implementation, the desired objectives were simplicity, avoiding redundant (and potentially asynchronous) information, and minimal code modification. The following are three options that were considered.

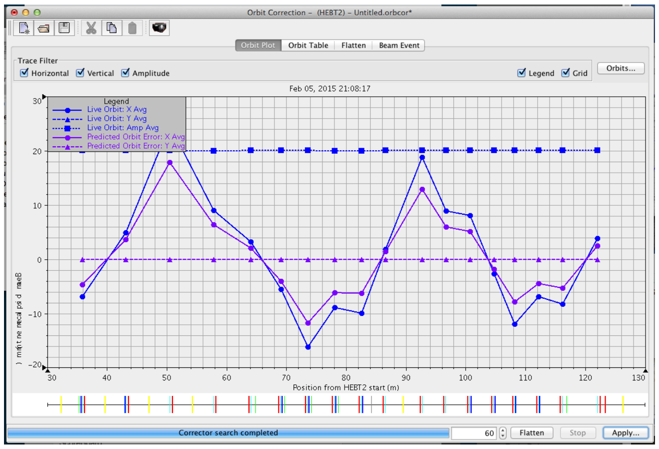
Weights could be attached as an additional objective and passed all the way through the Solver, accounted for by each algorithm. This was quickly done away with as it introduced unnecessary complexity and would alter parts of Open XAL outside of the application. A more reasonable approach was to introduce a weighting field into the `BpmRecord` class, used to persist information about the measured values in a BPM. Though this was more appropriate, there were two problems: a) the information would be redundant and ran the risk of becoming out of sync with other locations where the weighing information was stored and b) the nature of the overloaded constructors introduced repeated modifications and messy code.

Finally the developers decided that an array of the weights of the BPMs would be stored alongside their positions in the `Orbit` class (as well as the similar `BeamExcursion` class). The weights would be passed along until they reach the calculation of the rms displacement. The change in behaviour is simply a multiplication of each displacement with its weight while the rms is being calculated. The introduction to the user interface included the addition of a single column in the BMP table, with default weights set to 1.0. This was the most satisfactory implementation according to the desired objectives.

To determine the validity of this implementation, a series of rudimentary tests were performed using the test accelerator description included in Open XAL (`./core/test/resources/config/main.xal`) and the MEBT-HEBT combo sequence. Dramatic weights were put on various BPMs; the beam was flattened and then compared to the



(a) Unweighted orbit correction



(b) Underweighted last three beam position monitors (BPMs)

FIG. 1: The qualitative results of a single test of orbit correction with different BPM weighting values. The top figure shows an unweighted scenario (all values set to 1.0) and the bottom figure shows a scenario in which the final three BPMs are set to 0.01, very low weights. A difference in the expected orbit can be seen along the beam path. The direction of the change in displacement is not as expected and further testing and verification is recommended.

unweighted results. Qualitatively, the tests were successful. For example, reducing the weight of the final two BMPs significantly altered its exit angle. Dramatically increasing the weight of a rather deviated BPM brought the beam very close to centred for that one position, while offsetting the adjacent BMPs. Setting all the weights to 1.0 did not affect the flattening compared to the previous implementation.1

Future testing is required to further validate this method. A fully developed suite for regression testing would be desirable. Unit tests would also be appropriate for much of the code, for example, in the selection of usable correctors or in the optimization of a simple test case.

VI. CONCLUSION

According to rudimentary testing, the implementation of these changes was successful. The decision to store weighing information in the Orbit object and to modify the rms calculation minimized disruptions to the current code while still achieving the desired behaviour. This brings the existing orbit correction application closer to complete fulfillment of the presented requirements.

There is still a significant amount of work that can be done to further improve the program. First, formalized testing is vital for verification the algorithm, regardless of responder weighting. Second, the addition of more algorithms to the solver could improve the accuracy of the solution or the speed at which it is found; linear algebra-based solutions could be implemented as well. Finally, minor changes could perhaps reveal the calculated transfer matrix, a potentially very useful feature.

Most importantly, further investigation into the behaviour of this application is recommended. It has proven to be a powerful tool at different facilities and could be widely used at TRIUMF, with or without further modifications.

VII. ACKNOWLEDGEMENTS

I would like to thank my supervisors Ian Mitchell and Meghan Allen for their guidance and support in this project, Fred Jones and Yu-Chiu Chao for providing me this opportunity and sharing their knowledge, and the Open XAL developers for donating their time.

[1] GE Forsythe and Th S Motzkin. Asymptotic properties of the optimum gradient method. In BULLETIN OF THE AMERICAN MATHEMATICAL SOCIETY, volume 57, pages 183–183. AMER MATHEMATICAL SOC 201 CHARLES ST, PROVIDENCE, RI 02940-2213, 1951.

[2] John A Nelder and Roger Mead. A simplex method for function minimization. The computer journal, 7(4):308–313, 1965.

[3] Robin Newhouse and Tom Pelaia. Openxal orbit correction user manual, 2015. in progress.

[4] Tom Pelaia. private communication.

[5] Lloyd N Trefethen and David Bau III. Numerical linear algebra, volume 50. Siam, 1997.

[6] Greg White, Tom Himel, and Hamid Shoaee. A hybrid numerical method for orbit correction. In Particle Accelerator Conference, 1997. Proceedings of the 1997, volume 2, pages 2425–2427. IEEE, 1997.