Proceedings of the 1999 Particle Accelerator Conference, New York, 1999

A Free Interactive Matching Program*

J.-F. Ostiguy†, Fermi National Laboratory, Batavia, IL

Abstract

For physicists and engineers involved in the design and analysis of beamlines (transfer lines or insertions) the lattice function matching problem is central and can be time-consuming because it involves constrained nonlinear optimization. For such problems convergence can be difficult to obtain in general without expert human intervention. Over the years, powerful codes have been developed to assist beamline designers. The canonical example is MAD (Methodical Accelerator Design) developed at CERN by Christophe Iselin[3]. MAD, through a specialized command language, allows one to solve a wide variety of problems, including matching problems. Although in principle, the MAD command interpreter can be run interactively, in practice the solution of a matching problem involves a sequence of independent trial runs. Unfortunately, but perhaps not surprisingly, there still exists relatively few tools exploiting the resources offered by modern environments to assist lattice designer with this routine and repetitive task. In this paper, we describe a fully interactive lattice matching program, written in C++ and assembled using freely available software components. An important feature of the code is that the evolution of the lattice functions during the nonlinear iterative process can be graphically monitored in real time; the user can dynamically interrupt the iterations at will to introduce new variables, freeze existing ones into their current state and/or modify constraints. The program runs under both UNIX and Windows NT.

1 INTRODUCTION

Until just a few years ago, research software was difficult to localize and the distribution mechanisms were inefficient, making it difficult to build upon work done by others. The internet has dramatically altered this state of affairs. Before attacking a problem, it is now the norm to acquire and study existing source code. In this paper, I describe the BeamLine Interactive Matching Program (BLIMP), an interactive lattice design application assembled with various freely available software components. The objective is not to compete with commercial products, but rather to provide an application that can be modified and adapted to meet specialized needs. There are few available non-commercial interactive applications to perform beamline design. A well-known example is TRACE3D, which, despite being written in Fortran more than twenty years ago and adapted for interactive usage around 1988 is still widely used.

The design goals for BLIMP were the following: (1) given a nominal description of a beamline, allow a user to specify all aspects of a matching problem interactively (2) provide graphical feedback and allow the user to dynamically interrupt a nonlinear iteration to change the state of variables and constraints. As it stands now, BLIMP is still work in progress; nevertheless, basic features have been implemented and are fully functional.

2 THE MATCHING PROBLEM

The matching problem is a common lattice design problem. It can be simply stated as follows: given a beamline and a set of lattice functions specified at one extremity, determine the strength and/or longitudinal position of beamline elements necessary for the lattice functions to assume certain specified values at one or more distinct locations. The problem arises typically in the following situations: (a) a beamline is needed to transfer beam from one circular machine to another (b) a beamline with special optical properties is to be inserted into the regular lattice of a circular machine without perturbing the region lying outside of it.

In most situations of practical importance, horizontal and vertical motion are decoupled and a beamline is to first order, completely characterized by a set of ten quantities: $\beta_{x,y}$, $\alpha_{x,y}$, $\mu_{x,y}$, $\eta_{x,y}$ and $\eta'_{x,y}$, where β and α are the familiar Courant-Snyder functions, μ is the phase advance and η and η' are respectively the dispersion and its derivative with respect to the longitudinal coordinate. The effect of a mismatch in η' is often ignored; it is also common not to constrain the phase advance.

3 CODE STRUCTURE

BLIMP is written in ANSI standard C++ and makes use of the Standard Template Library. Variables are defined independently of basic beamline elements and can in principle involve arbitrary linear combinations of element strengths, making possible the definition of families of elements sharing a common power source. The user can dynamically define both local and global constraints. Typically, local constraints involve equalities while global constraints involve inequalities (e.g. β function smaller than a prescribed maximum). Figures 1 and 2 are screen shots of the user interface. BLIMP has been put together by using freely available software components which are now briefly described.

3.1 MXYZTPLK/BEAMLINE LIBRARIES

The MXYZTPLK and Beamline Libraries, authored by Leo Michelotti [1, 2], have been under development in the Beam Physics department since 1989. MXYZTPLK is a stand-alone library of C++ classes for performing automatic differentiation and differential algebra. In an nutshell, automatic differentiation is the systematic application of Liebnitz's chain rule to evaluate derivatives of ar-

* Work supported by the US Department of Energy.
† Email: ostiguy@fnal.gov

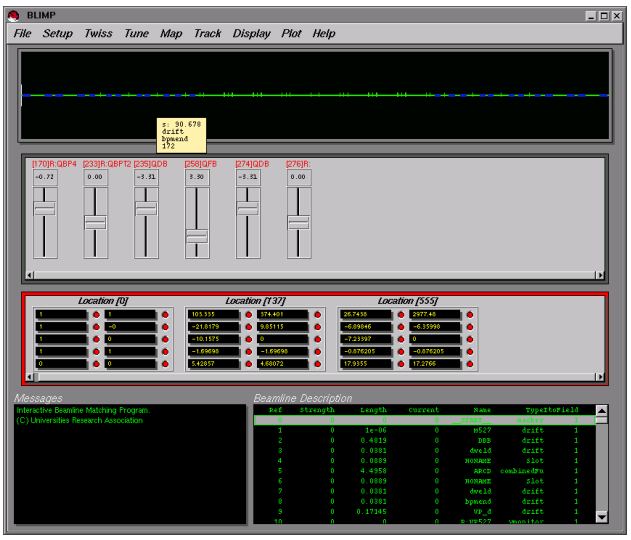

Figure 1: BLIMP user interface. The top window is a display of the beamline. The user can grab an element and move it interactively. In the second window, a sliding cursor is displayed for each variable that has been defined. Each variable can have constrained limits. The third window shows local constraints. Each of these constraints is editable. The bottom right window is a text browser that contains a description of the beamline elements. The user uses this window to select elements (variables) or locations (constraints).

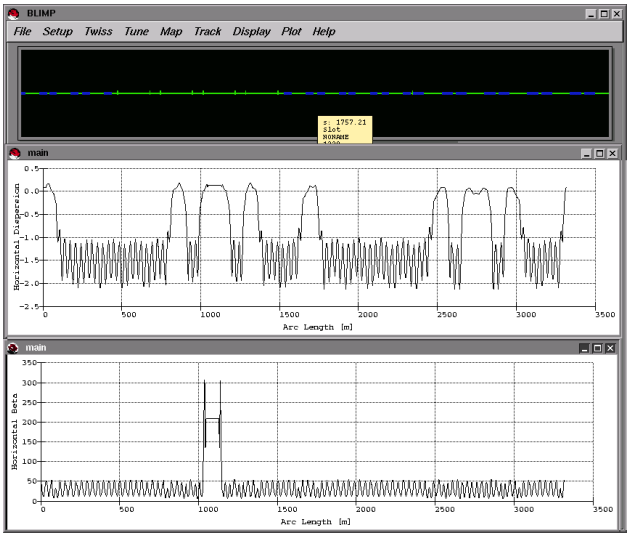

Figure 2: BLIMP displaying lattice functions. The plots are dynamically connected to the variable sliding cursors; changes are reflected in real time in the plot window(s).

bitrary order to machine precision. The Beamline class library built on top of MXYZTPLK, is a rich set of classes supporting lattice related calculations. Beamlines are represented by doubly linked lists whose nodes can either point to other beamlines or to basic elements such as dipoles, quadrupoles, RF cavities etc. Beamlines can be edited, concatenated, cloned, flattened (i.e. no hierarchical structure) etc. Most quantities of interest to accelerator physicists can be computed, including lattice functions, dispersion and chromaticity. Both field and alignment errors can be included if necessary. Maps of arbitrary order can in principle be computed to machine precision in either 4-dimensional (i.e. transverse) phase space or full 6-dimensional phase space. The desire to compute high order maps actually provided much of the motivation for developing the automatic differentiation library. BLIMP uses functionality from the Beamline class library to compute lattice functions, track individual particles or distributions and compute maps.

3.2 NONLINEAR OPTIMIZER

Numerical nonlinear optimization is a vast and highly specialized field. Nevertheless, problems can generally classified according to (1) whether or not the objective function can be expressed as a continuous, differentiable function of the independent variables and (2) the nature of the external constraints that need to be enforced, if any. For matching problems, the objective function are usually differentiable functions of the elements strengths and positions. In that case, variants of the gradient and Newton methods are most efficient. The Newton method has the advantage of quadratic convergence if the extremum is sufficiently close; the rate of convergence for gradient methods tends to be less favorable. However, because it is typically more expensive to compute and invert a Hessian matrix than to compute a gradient vector, a common strategy is to start with a gradient iteration and switch to a Newton iteration only within close proximity of the extremum. Constraints are most easily handled by adding penalty terms to the objective function. These penalty terms must obviously be differentiable; quadratic terms are useful for equality constraints while exponential terms can be used for inequality constraints. When inequality constraints apply to independent variables, a useful technique is to use an inverse trigonometric transformation and to consider the transformed variables as free.

At the moment, BLIMP uses the facilities of the MINUIT library from CERN [4], a good general purpose optimization library. It supports the optimization strategies described above and is freely available for research institutions. Unfortunately MINUIT suffers from various limitations associated with its Fortran heritage. Among the most problematic issues are the following: all I/O involves the Fortran I/O subsystem which cannot be mixed with the C/C++ I/O in a portable way; the objective function must be passed to the library as a static function and there is therefore no straightforward way of using functors objects. The BLIMP optimization code is encapsulated into an `Optimizer` class; this should allow an alternative to MINUIT to be substituted with minimal side effects.

3.3 GRAPHICAL USER INTERFACE

The choice of a user interface toolkit has been driven by two requirements: (1) object orientation and (2) need for portability between various flavors of UNIX and Windows NT. The Fast Light Toolkit (FLTK) [5] satisfies both requirements and is available under the terms of the GNU Public Library License. FLTK also provide support for OpenGL (or MESA, a free compatible alternative).BLIMP takes advantage of the facilities offered by OpenGL to efficiently display the beamline at different scales.

3.4 PLOTTING

BLIMP uses the SciPlot scientific plot widget[6]. SciPlot was written for the Xt toolkit and therefore uses an event model that is incompatible with FLTK. This difficulty was circumvented by running plot widgets in separate threads. At the moment, this is the only part of BLIMP that does not compile under Windows NT without modifications. One interesting aspect of the plotting functionality in BLIMP is that all user interface control elements are dynamically connected to the various plots, allowing a user to dynamically observe the sensitivity of a solution to small parameter variations, or to explore the parameter space before attempting a non-linear optimization.

4 APPLICATIONS

We now describe two applications that motivated the development of BLIMP.

4.1 PHASE TROMBONE

The Fermilab Recycler ring is a new machine for antiprotons accumulation and recycling scheduled to be commissioned in the spring of 1999. The Recycler has the distinction of being the first machine to make large scale utilization of permanent magnet technology. The machine operates at fixed energy of 8 GeV with a lattice based on fixed field combined function magnets. The tune of the machine is adjusted by varying nine electromagnetic quadrupoles grouped in five symmetric families within a region where $\eta_{x,y} = \eta'_{x,y} = 0$. Four hard constraints must be met i.e. at the symmetry point $\alpha_{x,y} = 0$ and the two phase advances set to the desired values; an additional softer requirement is to prevent the beta functions from exceeding a maximum value. Since this region is part of a matched insertion, the lattice functions outside the insertion remain unperturbed when the tune is adjusted. The maximum tuning range is approximately $\pm 1/2$. Since the settings of the quadrupole families is different for each tune, it is anticipated that BLIMP will be useful to both to adjust and diagnose the phase trombone.

4.2 LOW BETA INSERTION

In a low-beta insertion, the objective is to use a pair of quadrupole doublets or triplets to focus counter-circulating beams into a very small size interaction region. In general, the insertion has to match the lattice functions of the ring at both extremities; the phase advance is unconstrained. At the interaction point, $\beta_{x,y}$ must assume specified values and the beam envelope must go through a minimum i.e. $\alpha'_{x,y} = 0$. It is also often required for the dispersion to be as small as possible and one usually demands $\eta_{x,y} = 0$. Constraining η' may also be desirable. The result is 24 local constraints. In practice, low insertions are symmetric and one can concentrate one one-half of the insertion, reducing the number of constraints to 16, possibly 14 if η' is ignored. Global constraints are usually introduced to limit the amplitude of the beta functions inside the high gradient quadrupoles.

Low beta insertions are notoriously nonlinear. Without experience, it is difficult for a novice to find a satisfactory solution and interactivity is certainly no substitute for experience. However, the ability to quickly experiment with different strategies and stop the iterations dynamically is proving to be a significant advantage.

5 CONCLUSION

BLIMP is still work in progress, although it is certainly already useful as it stands. In the immediate future, the program will acquire the ability to read and write in a "standard" beamline specification format. Even though the format used at this moment is application specific it is a very simple matter to describe a simple beamline with a few dozen elements. A beamline editor allowing the user to specify or modify beamlines interactively would also be an interesting improvement. All source code should eventually become freely available to the accelerator community, at least for non-commercial use. Please contact the author for further informations.

6 REFERENCES

[1] L. Michelotti, "MXYZPLTK Version 3.1 User's Guide: A C++ Library for Differential Algebra", Fermilab Publication FN 535-REV, October 1995

[2] L. Michelotti, "MXYZPLTK and Beamline: C++ Objects for Beam Physics", Advanced Beam Dynamics Workshop on Effects of Errors in Accelerators, their Diagnosis and Correction, AIP Conf. Proceedings No 225, 1992

[3] C. Iselin and H, Grote, "The MAD Program (Methodical Accelerator Design), User's Reference Manual, CERN/SL/90-13(AP), Geneva, Switzerland

[4] F. James, "MINUIT Minimization Package Reference Manual Version 94.1", CERN Program Library D506, Computing and Networks Division CERN Geneva, Switzerland

[5] Information about FLTK is available at the following URL: `www.fltk.org`

[6] Information about the SciPlot widget is available at the following URL: `www.ae.utexas.edu/~rwmcm/SciPlot.html`

[7] Fermilab Recycler Ring Technical Design Report Revision 1.2, November 1996