MAD VERSION 9

F.C. Iselin, J.M. Jowett, J. Pancin, CERN, Geneva, Switzerland
A. Adelmann, PSI, Villigen, Switzerland

Abstract

The program MAD is widely used for accelerator design and beam dynamics studies. For many years, its input language has been the nearest thing to a world-wide standard for describing accelerator structures. The new Version 9 is a complete rewrite using a systematic object-oriented methodology based on the CLASSIC classes [2] for accelerator physics. It provides many improvements over the previous MAD Version 8. These include: (i) support for multiple beam-lines simultaneously, facilitating, for example, matching constraints that couple the two rings of a two-ring collider, (ii) much improved Lie-algebraic map calculations, (iii) a uniform method and format for exchanging many kinds of structured data with other programs, (iv) an improved and more consistent input language. In addition, we report on a parallel 3D Poisson field solver for space charge calculations in high intensity particle beams. Applied to the PSI injector cyclotron, this shows the general nature of MAD Version 9 as a state-of- the-art problem-solving environment. We describe the current status of the program and how to get it, outline future plans and illustrate some of the new features.

1 INTRODUCTION

We shall describe the new MAD Version 9 (MAD9) in contradistinction with MAD Version 8 (MAD8) [1].

1.1 Implementation

Whereas MAD8 and its predecessors were written in the Fortran language (extended somewhat by dynamic memory management, etc.), MAD9 is an object-oriented program written in standard C++ using the design pattern paradigm [5] for the overall architecture. It is divided into two main parts: (i) the CLASSIC framework which embodies the accelerator physics (ii) the MAD9 (driver) program containing a few non-physics classes like parser, etc.

1.2 Deep and shallow extensibility

Experience has shown that the idea of a universal accelerator physics program is a chimera [3]. With this in mind, we have sought to provide *two paths* for extension of MAD9's existing capabilities.

Programming MAD9 We need not vaunt the benefits of object-oriented programming. This can be exploited in the *deep mode* where extensions to MAD9 are undertaken in a disciplined manner by designing and writing additional classes and incorporating them seamlessly into the existing framework. This task requires a background in computational accelerator physics, object-oriented design methodologies and a profound C++ language knowledge.

MAD9 as a software component The design of MAD9 also gives considerable attention to clear programming structures as perceived by the user of the MAD9 language. All essential output comes in a uniform self-describing ASCII table format that is easily read into many other programs and environments (e.g., spreadsheets, Mathematica and, not least, control applications). Two-way interaction (in which the external environment sends commands to a MAD9 process) opens up the possibility of subsuming the operation of MAD9 into function calls in another environment. Ths *shallow mode* allows many new capabilities to be added without modifying MAD9 itself.

We feel that the importance of this mode cannot be overemphasised: it has been motivated by practical experience with MAD8, which went only a part of the way to providing such functionality.

Development up till now has been mostly concerned with the traditional console-style application, driven by a stream of input data and commands and producing output in files. We expect that most users will adopt higher level and/or graphic interfaces as they become available.

A primary motivation here is to interdict the accretion of modules imported from other programs, a major source of overlapping and inconsistent calculations. In case of urgent need to implement some feature, it will always be more practical to do it the shallow way than to embark on hasty changes to the core program.

2 THE MAD9 LANGUAGE

Broadly speaking, the MAD9 language supports two main idioms: the description of beam lines or accelerators and the specification of calculations to be done on them, the "Action Commands".

2.1 Beam line description

The MAD8 input language or Standard Input Format (SIF) [9] is widely accepted as a means of describing beam-line structures. Accordingly, the changes made in this part of the language have been kept to a minimum.

Many of them have been made for the sake of syntactic consistency or to facilitate constructions that were awkward in the MAD8 language. In the former category we may note that abbreviated forms of commands are no longer accepted and each input statement must end with a semicolon. The latter category includes more interesting innovations such as the ability to specify a whole set of

multipole field components as a "vector", or the ability to make a beam line common to two or more larger structures.

2.2 *Action commands*

Here the changes are more significant. The **USE** command of MAD8, that selected the beam line to be operated upon (e.g., to have its linear optics computed by the **TWISS** command), has disappeared. Instead, commands like

```
injBeam:   Beam, particle=proton, energy=400;
injOptics: Twiss, line=Ring1, beam=injBeam;
```

are used to define a named beam **injBeam** and a named table **injOptics** which contains the optical functions computed for that beam in a previously defined ring **Ring1**. The beam-line description unambiguously defines the fields in the elements *independently of any beam* (although an arbitrary constant P0 is used to convert units). Thus MAD9 respects the physical reality that a given beam line can simultaneously have different optics for different beams.

3 PRESENT FEATURES

Users of MAD8 will find many improved versions of familiar features in MAD9. Optical calculations are more complete and provide results in more convenient forms. Matching can be done on a collection of output tables from geometry survey to transfer matrix elements. Problems coupling different beam lines can be treated. Some Lie algebraic map and tracking methods are implemented. Specification of multipole and alignment errors is much simplified, especially for machines like the LHC. Synchrotron radiation effects have not been implemented yet.

3.1 *Aperture calculations*

The geometrical aperture calculation is a new feature in MAD9 , designed to replace an independent program [4] previously used for LHC studies It computes the maximum allowed aperture of primary collimators in terms of the RMS beam size at any desired position in the machine. The local maximum aperture is obtained by calculating the largest inscribed secondary halo inside the vacuum chamber, as a function of the vacuum chamber shape and size and various tolerances [4] given as input. The implementation in MAD9 allows *aperture constraints to be used in matching* processes.

The aperture module interpolates the Twiss functions inside each selected element. It calculates the beam halo taking into account the size of the primary and secondary collimators. The dispersion, closed orbit and mechanical tolerances permit an estimate of the displacement of the beam at any point. Combining the two, the largest possible secondary halo inscribed in the vacuum chamber may be calculated. The user can choose any shape for the halo and the beam screen. The more exotic ones have to be entered as a list of co-ordinate pairs.

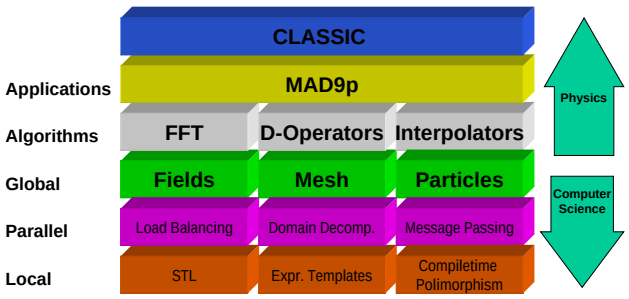

Figure 1: (colour) MAD9P structural overview

Its main limitation at present is the use of uncoupled Twiss functions that, while quite adequate for the case of the LHC lattice, may be inaccurate for strong coupling.

4 PARALLEL 3D TRACKER WITH SPACE CHARGE

To distinguish this special version we will use the name MAD9p. At present MAD9P is not included in the official CERN MAD9 distribution.

The driving force behind this effort is the better understanding of the high intensity particle beams in the accelerator complex of the Paul Scherrer Institut [11]. Theoretical understanding of these effects is just sufficient to understand qualitatively the principal beam behaviour but clearly lacks the depth needed to make quantitative predictions. For this research project a 3D code is being developed to simulate beam behaviour dominated by space charge.

4.1 *Models and implemented procedures*

For all calculations, we assume a collision-less system and no special *a priori* charge density distribution. Radiation and beam pipe heating are not considered. A collimator simply removes particles from the computing domain. No secondary effects are considered so far.

The space charge calculation is integrated into the tracker part of MAD9P. The tracker and the field solver have been parallelised using the POOMA (Parallel Object Oriented Methods And Applications) framework [12]. MAD9P and all other components used are based on object oriented design philosophies (Framework and Design Pattern). There are no *a priori* restrictions on the number of particles or number of workstations to be used. See Figure 1 for an rough overview.

Often the Hamiltonian can be written as a sum of two parts, $H = H_{ext} + H_{sc}$, in our case the Hamiltonian for the external forces and the space charge. Such a form is ideally suited for the application of symplectic split-operator methods [6].

More generally, consider a Hamiltonian that can be written as a sum of two parts $H = H_1 + H_2$, where each part,

separately, can be solved exactly or to some desired accuracy or order.

In our case $H_1 \equiv H_{ext}$ includes the external fields computed by CLASSIC to any desired order using Lie algebraic methods combined with Truncated Power Series (TPS).

The second term, $H_2 \equiv H_{sc}$, corresponds to the space charge fields. At present we use a parallel 3D Fast Fourier-based Poisson solver assuming open boundary conditions.

Given the corresponding maps, $\mathcal{M}_{ext}$ and $\mathcal{M}_{sc}$, (1) is accurate through 2nd order in the independent variable τ [7]:

$$\mathcal{M} = \mathcal{M}_{ext}(\tau/2)\mathcal{M}_{sc}(\tau)\mathcal{M}_{ext}(\tau/2) \qquad (1)$$

This means: track the particle through the first part of the element, apply a space charge kick over the whole element and finally track through the second part of the element.

4.2 Preliminary results and future

New Commands and Attributes are provided to do the parallel tracking and space charge calculation. Briefly:

DISTRIBUTION defines a particle distribution in 6D (Gaussian) or reads in a distribution from a file,

TRACKDIST tracks a distribution of particles,

STATMARKER saves or displays RMS beam quantities and emittance,

PARTMARKER saves or displays 6D particle data and E-field magnitude and direction,

COLLIMATORS removes particles which either hit a rectangular collimator or the beam pipe. Particle positions and IDs are saved for later (loss analysis).

SCKICK defines the number of space charge kicks in each element,

SCFACT varies the magnitude of the space charge kick in order to simulate beam neutralisation,

FIELDSOLVER this attribute on the BEAM command selects one of the field solvers

A parallel post-processing tool MAD9PVISscan be used for various post-processing tasks including movie-making. Data suitable for GNUPLOT can be exported either directly by MAD9P or MAD9PVIS for plots of RMS beam sizes, emittance or particle loss statistics.

MAD9P compresses and saves data in files using a SMP (symmetric multi-processor) -like approach.

The code has passed well a set of test cases, namely 'expanding sphere' and 'FODO channel' with acceleration. All test cases were checked against a modified version of MaryLie [8] which includes methods for space charge calculation. We are now going forward to model the PSI Injector II cyclotron including the buncher section.

Future Development A serial version of a tree-based (gridless) Poisson field solver is already available and we are in the process of finishing the parallel implementation. The availability of two different field solvers will be a unique feature and allow us to check the physical meaning of results obtained.

Image charges will be added using a pre-calculated capacitance matrix, a geometry factor of the beam pipe [10].

5 STATUS OF MAD9

MAD9 is an ambitious project, attempting to synthesise many years of experience with the earlier versions into a new program that provides a much sounder foundation for future development. We stress, however, that the project is incomplete: both the present program and its documentation contain bugs and lacunæ. Some calculations need to be speeded up. Yet the same is true of MAD8 (and many other programs!) and we consider that the chief advantage of the switch to modern software technologies is that MAD9 will soon, if it has not already, overtake MAD8 in software quality, maintainability and reusability.

MAD9 is already the basic optical tool for the LHC project. Extension, testing and debugging of the program continue within this framework and growing collaborations. The source of all information, including the code and users' guide, is [1].

REFERENCES

[1] The MAD Home Page, http://wwwslap.cern.ch/mad/

[2] F.C. Iselin, Proc. Int. Computational Accelerator Physics Conf., Monterey, Sept. 1998.

[3] Senses 2., 3.b–d, OED, 2nd Edn., 1989, are apt.

[4] J.B. Jeanneret and R. Ostojic, CERN LHC Project Note 111 (1997).

[5] E.Gamma, R. Helm, R. Johnson and J. Vlissides *Design Patterns* Addison-Wesley 1994

[6] E. Forest, et al., Phys. Lett. A 158 , 99 (1991).

[7] E. Forest and R. Ruth, Physica D43 , 105 (1990).

[8] R. Ryne, private communication.

[9] D.C. Carey and F.C. Iselin, Snowmass Summer Study 1984

[10] K. Nabors and J. White, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, **10** pp. 1447–1459, November 1991.

[11] URL: http://www.psi.ch

[12] J.C. Cummings and W.F. Humphrey, 8th SIAM Conf. Parallel Processing for Scientific Computing, 1997.