UPDATE ON MAD-X AND FUTURE PLANS

F. Schmidt (MAD-X custodian*), CERN, Geneva, Switzerland

Abstract

After a intense and hectic code development during the LHC design phase the MAD-X [1] program (Methodical Accelerator Design – Version X) is going through a period of code consolidation. To this end the development on the core has been frozen and most efforts are concerned with a solid debugging in view of a trustworthy production version for the LHC commissioning. On the other hand, the demand on further code development from the LHC pre-accelerators and CLIC are dealt with PTC [2] related parts of the code where the implementation is in full swing. Having reached a mature state of the code the question arises what kind of future can be envisaged for MAD-X.

INTRODUCTION

During the year 2000 the MAD-X project had been started and a first version had been released in June 2002 [1]. Since then all parts of the code have been adapted frequently to deal with constantly changing requirements needed for the LHC design. Since some time now we have shifted our focus to the upcoming LHC commissioning. Since MAD-X will be playing a relevant role in the LHC operation.

Module	Description	Keeper
MAD-X C Core	Maintenance & Debug	H. Grote
APERTURE [3]	Modeling LHC Aperture	J.B. Jeanneret
C6T [4]	SixTrack Converter	F. Schmidt
CORORBIT	Orbit Correction	W. Herr
DYNAP	Tracking Postproc.	F. Zimmermann
EMIT	Emittance, Radiation	R. Tomas
ERROR	Error Assignment	W. Herr
IBS	Intra-Beam Scattering	F. Zimmermann
MAKETHIN	Thin lens Converter	H. Burkhardt
MATCH	Matching Procedures	E. Laface
PLOT	Mad-X Plotting	R. de Maria (interim)
PTC [2]	PTC proper	É Forest KEK
PTC_NORMAL	Normal Form Coeff.	F. Schmidt
PTC_TRACK	Thick lens Lattice Track	V. Kapin ITEP (RU)
PTC_TWISS	Ripken Optics Para.	J.L. Nougaret
SODD [5]	Resonance Comp.	F. Schmidt
SURVEY	Machine Survey	F. Tekker
SXF [6]	Stand. eXchange Format	N. Malitsky BNL
TOUSCHEK	Touschek Effect	C. Milardi IFNL/LNF F. Zimmermann
TWISS	Classical Optics Para.	–
THINTRACK	Thin lens Lattice Track	Y. Sun

Table 1: Module Keepers, People in RED are collaborators from outside CERN, corresponding laboratories in BLUE.

* MAD-X Module keepers see Tab.1

During the years 2008/2009 we have been working on a consolidation of the code such that we can provide a trustworthy MAD-X production version for the LHC commissioning. In particular, since we are presently developing an on-line model [7] of LHC that is based on MAD-X.

On the other hand, there are new demands from other applications notably from the LHC upgrade program, the studies on the LHC pre-accelerators and the CLIC project. Presently, these demands concentrate on better performing modules based on PTC. These developments have been continued and they are not interfering with the consolidation effort for the LHC commissioning.

In this report the key features of the MAD-X are reviewed followed by the most relevant recent MAD-X highlights. The consolidation phase was used to further professionalize the code maintenance which is dealt with in some detail in the next chapter. Besides several outdated features of MAD-X, due to the fact that it is a successor to MAD8, we had to use shortcuts to allow for a gluing of MAD-X with PTC. These shortcuts cannot easily be undone and pose limitations for a further development of the code as a whole. In particular since we will need MAD-X proper in its present state for the LHC commissioning effort. This report closes with a discussion of how one could modernize the code and achieve a better integration of PTC with MAD-X.

KEY FEATURES

The status of MAD-X can best be understood by looking at the design goals of MAD-X followed by a description of what PTC is and how it is connected to MAD-X. From the start we have decided to distribute the work of the code development and maintenance to a large group of module keepers (see Tab. 1) with a code custodian orchestrating their efforts. The job of the custodian is to direct the development of the code and to guarantee its integrity with the goal to provide a code that can handle all requirements for the LHC design and the commissioning alike.

Design Goals

The task at the time was to provide a code in a very short period to allow the design work for the LHC. No grand scheme could be attempted instead we made use of most of the well debugged MAD8 source code (Fortran77) including the traditional MAD8 strengths like sequence editing, matching, plotting, closed-orbit and error routines. However, MAD8 also included out-dated techniques and some feature were flawed if not plain wrong. Those features had to be eliminated: e.g. thicklens non-symplectic tracking

was replaced by a symplectic thintrack module. Another major step has been the replacement of Lie-Algebra by the more flexible Differential Algebra Map [8] and Normal Form techniques [9]. Memory management via C code was used to replace CERN-made Zebra (Fortran77) and the concept of link-lists for the elements in an accelerator have been adopted. A better modular structure of the code was attempted with C interfaces to isolated modules in Fortran or C. The input language (parser in C) was chosen close to MAD9 language and includes FLOW statements like (while, if, macro etc). Deferred expressions are invoked by the ":=" symbol rather than the traditional one "=". Moreover, a complete pattern matching has been implemented.

Through out all the MAD-X modules output data are written to self-described TFS Format with a two-way TFS-SDDS [10] converter, both formats are self-described data sets. For a complex code like MAD-X the version control system CVS has been an essential tool mandatory for a sound program evolution in particular when dealing with a large number of contributors.

The documentation is presented via the MAD-X web page (google search: madx cern). The user guide can be found via a HTML page. A keyword search and a PDF manual is supplied as well. On the MAD-X web page one can find supporting reports and minutes of the MAD-X meetings. One can download the source code and executables. Moreover, all examples can be found collected in the madx-examples CVS repository.

There have been 2 major objectives: a usable MAD-X code needed for the LHC design which has been delivered by July 2002. After years of development MAD-X has now been prepared for the LHC commissioning and the LHC on-line model for the end of 2009. It should also be mentioned that MAD-X also has to serve all LHC pre-accelerators. To this end a CVS repository CAMIF (Cern Accelerator Mad-X Input Files) has been worked out by CERN staff for all Cern machines as of the PS booster till the LHC and including the transfer lines. Last but not least MAD-X has become a major design code for CLIC in particular with the PTC extension.

PTC

The single most important change to MAD8 is the inclusion of PTC of É Forest [2] as an integral part of MAD-X. The code PTC is composed of two distinct parts (in Etienne's words):

A) The independent polymorphic library FPP [8, 11] which handles the operations on polymorphs. Polymorphs are capable of transforming real numbers into Taylor at execution. FPP also contains all the critical operation on Taylor maps: Normal Forms, various factorizations, etc

B) The code PTC proper is at heart a symplectic integrator with classical radiation added on top. It also moves around quadratic stochastic beam envelopes.

However PTC is a physics engine only. It offers hardly any tool of convenience in itself, except when absolutely mandatory like a closed orbit searcher. PTC remains É Forest's independent (Fortran90) program and is linked to MAD-X via a converter. All modules based on PTC (also in Fortran90) have to be provided by the MAD-X team, e.g. ptc_track (symplectic thicklens), ptc_twiss and ptc_normal (responsibilities see Tab. 1). The design goal for the integration of the two codes has been: all accelerator elements of MAD-X should be treated in PTC and agree with MAD-X as long as the physics is correct with the added value that the elements in PTC are correctly described for any momentum deviation. Special emphasis is on a proper description of RBEND/SBEND, only the later is truly an element of MAD-X. This concept may best be illustrated by an example. At the PAC03 [12] it was demonstrated that MAD (green dots) cannot handle delta-p dependence very well. For this cyclotron I have prepared a MAD-X run both executing a TWISS and PTC_TWISS command. The latter (blue dots) agrees perfectly with the theoretical prediction (red line).

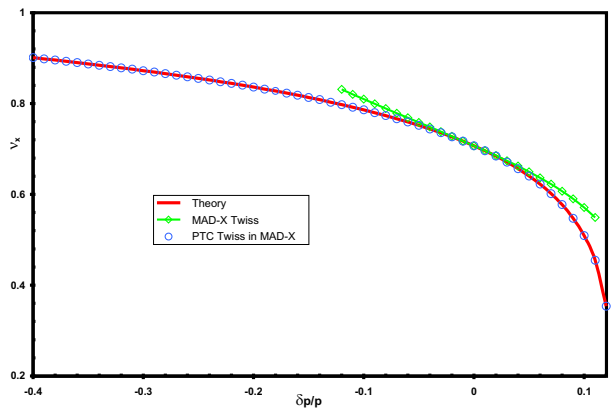

Figure 1: Simple cyclotron for MAD-X and PTC in comparison with theory.

This result is not a surprise to us but the very reason why we have complemented MAD-X with PTC. Please note, that the delta-p in this example is very large contrary to the LHC where it is so small that TWISS is quite sufficient for our purposes.

HIGHLIGHTS

MAD-X General Status

In 2008/2009 the MAD-X team has used the time for a consolidation of the MAD-X code: except for cosmetic changes the development of the core code has been stopped. Most efforts have been concentrated on a rigorous debugging (presently, September 2009, still on-going). In this mature state a major effort has been launched to maximize performance (culprits have been: table access and

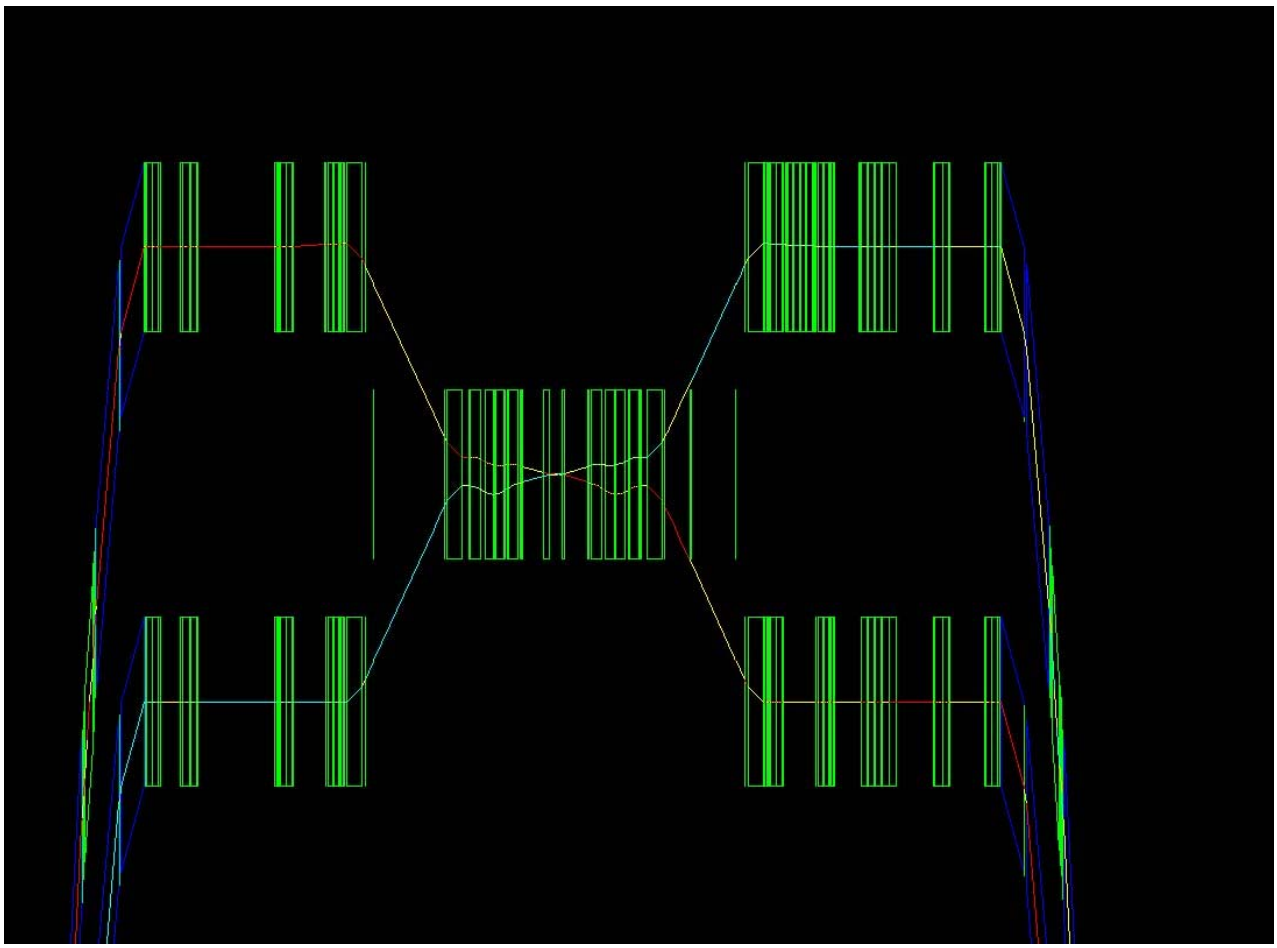

Figure 2: High Beta LHC Interaction Zone in the stand-alone PTC model.

database searches). As a result speed-up factors of 3-5 have been reported in LHC matching. One major change has been the suppressing of the highly controversial individual BV flag for separation dipoles in the common area used to distinguish between beams going in either direction. Instead, we did change the LHC sequence file for the counter-clockwise beam (BEAM2). After this special effort we are confident that our goal has been reached to provide a trustworthy MAD-X program with all needed features for the upcoming LHC commissioning.

It should also be mentioned that MAD-X has become one of the codes that can communicate accelerator input between those codes via AML and the Universal Accelerator Parser [13].

Module Development

During the MAD-X consolidation phase the status of the original modules have been reviewed. The goal was to have all modules ready for the LHC commissioning.

Aperture considerations will be very relevant in the early stages of the LHC commissioning. To this end the aperture module was modified various times, both to add new features and to clean-up the coding.

Studies for CLIC has revealed that the standard TWISS module suffered from various shortcomings with respect to chromatic functions and delta-p dependence in general. A full solution of all these problems would require a re-write of the code for which we simply lack the manpower. Therefore, we decided to fix the worst problems, in particular some improvement for the chromatic functions and a numerical calculation of the chromaticity which was incorrect in presence of linear coupling. Unfortunately, we cannot simply replace this module with ptc_twiss since computing speed is relevant for large machines like LHC in particular in matching.

For CLIC studies at very large delta-p dependence there is no choice but using ptc_twiss which delivers the desired results under any condition. We have therefore assigned a module keeper to this module. The goals are:

a) Provide the same elaborate information as the standard TWISS module.

b) Adding additional parameters, in particular dispersion and momentum compaction in second or higher order both for rings and lines alike.

Unfortunately, the thintrack modules had relied on the results of the TWISS module. This had to be changed since these 2 modules are based on different Hamiltonian descriptions. The thintrack module is now fully self-contained.

Presently, we are just missing a few more module features: the noise element will be added and the treatment of radiation will be improved both in the thintrack module.

PTC Advancements

Up to now PTC was based on one DA package by M. Berz [14]. Recently, Etienne got in contact with Lingyun Yang [15]. His DA package is written in C++ requiring an elaborate interface in PTC. Now both DA packages co-exist in PTC. Results are identical and the C++ package is even slightly better in performance. It should also be noted that this new package is optimized for use with many variables which still needs to be fully exploited.

The combination of PTC with ORBIT [16], has been recently achieved and successfully applied for the J-PARC Main Ring to fight emittance growth by a skew resonance compensation [17].

Another major development has been the addition of spin routines to PTC. Applying it to MIT-Bates South Hall Ring [18] it could be demonstrated that spin resonances as calculated by PTC agree perfectly with the prediction of the linear response theory [19].

MAD-X in the LHC Control System

The LHC control system LSA [20] uses TWISS TFS output files from MAD-X for their controls during all the LHC cycle.

Similarly relevant is the LHC on-line model [7] to safeguard LHC operation. This on-line model is based almost exclusively on the MAD-X code. This system is being developed in close collaboration of the operation and accelerator physics groups at CERN, the OP and ABP group respectively.

Lastly, we are in the process of constructing a full 3D model of the LHC with the PTC code in stand-alone mode. A MAD-X run provides pieces of the machine that are placed in the virtual tunnel according to SURVEY information. The harmonics to high order are added to all thick elements, magnet assemblies are misaligned, and separately the magnets within them. A complete aperture model around the ring is also taken into account. In Fig. 2 one finds a blow up of one of the high beta interaction zones. By constructions this is really a single structure and the light blue and red tracks are particles being tracked in both directions. In the common area these particles are steered via crossing and separation schemes respectively and are brought into collision at the interaction point quite like in the real world.

CODE MAINTENANCE

Integrity

The code development set-up with a sizable number of module keepers requires a more rigorous testing to ensure code integrity. To this end we have set up an automatic system to test several examples for every MAD-X module. These tests are started automatically once a MAD-X CVS tag has been set. The tests are performed 3 times with the standard executable and 2 specially compiled executables. This is needed to find hidden run-time errors, un-initialized variables, out-off-bound arrays etc. In case of problems the module keepers are alarmed and asked to correct code and/or the examples, which are kept in a separate madx-examples CVS repository. We have adopted the concept to provide a production and a debug version, the latter being the latest and greatest but at the risk of the user. A production version for LHC commissioning requires a successful passing of the complete example testing.

Portability

The next relevant design goal is portability across the 3 relevant platforms LINUX, Windows and MAC-OSX. We have one responsible for each platform. Particular worrisome is the fact that we are dealing with a multi-language code which now also includes a C++ part. With the advent of free GNU Fortran compilers (g95 and gfortran) and several commercial ones (Intel ifort, Lahey lf95, Nag f95) we could transform the Fortran77 to Fortran90 and provide just one executable which combines MAD-X and PTC. Presently, the ifort compiler has become our workhorse since it has worked effortless on Linux and Windows (available for MAC) and it offers a significant performance boost compared to the other compilers.

Special Issues

Separate output buffers for combined Fortran/C code may lead to garbled-up output. This problem is taken care of by automatic python scripts that modify the code prior to compilation such that the corresponding buffer is flushed whenever C calls F or the reverse. To allow for 64 bit compilation the C part of the code needed some fixing. For portability reasons we provide just the 32 bit executables. Tools have been provided to facilitate memory leak detection. In regular intervals we have to check both Fortran and C parts and take action if too many leaks have been found. For portability on the various Linux distributions we need statically linked executables. Unfortunately, the rapid Linux evolution requires changes to the Makefile for every new Linux distribution. The link step will therefore require fiddling in most cases that have not yet been covered.

LIMITATIONS

- The MAD-X parser in C is by now very complex and difficult to modify.
- The TWISS module has reached its end-of-life status. It exists for performance reasons only.
- The PLOT module is also difficult to maintain.

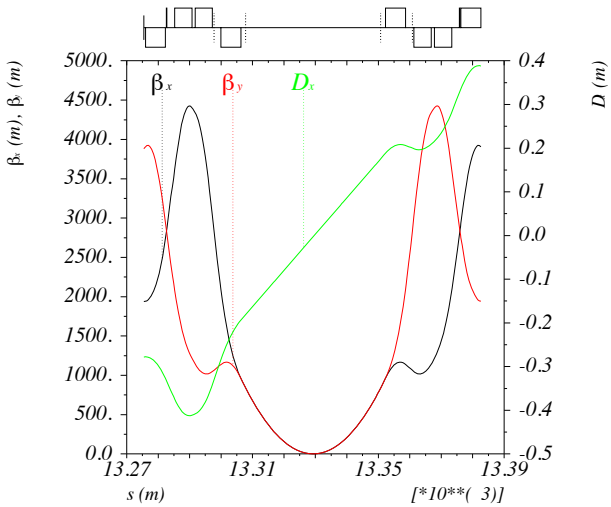

Figure 3: Plot example a high-beta LHC interaction zone.

This can best be explained with an plot example of a high-beta LHC interaction zone. It shows 3 Twiss parameters with 2 vertical axis, one at either side of the graph. Moreover, on the top of the graph the machine elements are symbolically placed at their proper s location in this case of the LHC. These kinds of complex plots are well suited for the design of accelerators. Any replacement of this module would require the exact same functionally. Trouble with the present module is that it is highly complex and difficult to fix, e.g. the invisible subscripts at the vertical axis should be modified. The interactive plot mode on Linux requires the libX11.a library to produce a statically linked executable which requires special installations for each new Linux distribution.

- The documentation is pretty old-fashioned and should be replaced by a professional tool.
- The main problem: MAD-X and PTC have separate structures, e.g. matching (MACROs needed) are extremely inefficient since in each step the PTC structure has to be recreated. In the present multi-language environment there is no clear path of how to overcome this problem.

PLANS

MAD-X has reached a mature status and can be used as is for the LHC commissioning. Minor modifications and bug fixes will continue.

PTC development can happily continue or speed-up to serve smaller machines and CLIC.

A truly integrated code would require a new start, i.e. something like MAD-11.

The design goals of MAD-11 as I see them are:

- MAD-X should be a subset of MAD-11 with exactly the same input language facilities, modules and features except for bugs.
- Parser, plot module and documentation should be build with external, well established and long lifetime tools.
- A single accelerator structure in memory both used for MAD and PTC is clearly needed.
- The big question is if such a goal can be reached in the present situation of a multi-language set-up rather than a monolith in a single computer language.
- Following the experience of the MAD-X project a reasonable estimate of the project duration is 5 years given appropriate resources and manpower. The goal would be to reach the same functionality as MAD-X but with a much better structure of the code.

REFERENCES

[1] H. Grote and F. Schmidt, PAC03, p. 3497, Portland, USA.
[2] É Forest, E. McIntosh and F. Schmidt, KEK Report 2002-3.
[3] I.K. Waarum, Sør-Trøndelag University College, Trondheim, Norway, 2004.
[4] F. Schmidt, CERN SL/94-56 (AP), (1994) (Rev. July 2008).
[5] F. Schmidt, CERN SL/Note 99-099 (AP).
[6] H. Grote, et al., RHIC/AP/155.
[7] I.V. Agapov et al., PAC07, p. 3384, Albuquerque, USA.
[8] É Forest, Y. Nogiwa and F. Schmidt, ICAP06, p. 191, Chamonix, France.
[9] M. Berz, É. Forest and J. Irwin, Part. Accel. 1989, Vol. 24, pp. 91-107.
[10] M. Borland, ICAP98, pp. 23-27, Monterey, USA.
[11] based on an idea of J. Bengtsson and on the DA packages of M. Berz, Lingyun Yang or others.
[12] D. Trbojevic et al., PAC03, p. 3405, Portland, USA.
[13] D. Sagan et al., EPAC06, p. 2278, Edinburgh, Scotland.
[14] M. Berz, Part. Accel. 1989, Vol. 24, pp. 109-124.
[15] Lingyun Yang, this conference.
[16] É Forest et al., discussed at the 2006 ICFA HB06 workshop.
[17] É Forest et al., to be published.
[18] É Forest and S.R. Mane, private communication.
[19] S.R. Mane, Nucl. Inst. Meth A 601 (2009), pp. 256-263.
[20] M. Lamont, et al, ICALEPCS07, p. 307, Knoxville, USA.