ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-WEPL101

STATUS OF MAD-X V5.09

R. De Maria*, L. Deniau, J. Dilly, J. M. Gray, A. Latina,
F. Schmidt, P. Skowronski, CERN, Geneva, Switzerland,
J. S. Berg, BNL†, Upton, New York, T. Gläßle, University of Tübingen, Tübingen, Germany

Abstract

MAD-X is a popular beam optics code used to design, model, and operate a large number of synchrotrons and linacs. In this paper, we present the features added in the most recent versions and improvements intended for future releases. Physics models have been added and improved to support the needs of the Future Circular Collider (FCC) and the Electron Ion Collider (EIC), regarding machine-detector interface, complex beamline layouts, and synchrotron radiation. More precise physics models have been implemented for some elements, and a complete set of exact coordinate frame transformations are now available. The tracking module has been extended to support frozen space-charge models. To improve interoperability with scientific ecosystems, MAD-X relies on the cpymad Python interface which offers fine-grained control of MAD-X simulations, exceeding the capabilities of the internal MAD-X language.

INTRODUCTION

MAD-X [1] is ending the second decade from its conception and continues to be used and developed in the context of accelerator design and operation. It is the reference tool to prepare optics configurations and optics models for the LHC collider and its injectors. It is currently used in the design of the Future Circular Collider (FCC) [2] and the Electron Ion Collider (EIC) [3], among others.

MAD-X offers a combination of the following features:

- large user base and common experience,
- well-established, large and stable feature set,
- flexible and convenient scripting language,
- good performance, no dependencies,
- regular support and frequent releases for active projects,

with fast Twiss calculations used for optics optimizations.

The cost of maintaining the internal structures, the technical debts, the pressure to add new features and grow the scopes, while retaining backward compatibility are the main challenges for the code.

The following sections will describe the main structure of the code, the recently added physics, and usability improvements.

* riccardo.de.maria@cern.ch

† This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy. The United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript or allow others to do so, for United States Government purposes.

Code structure

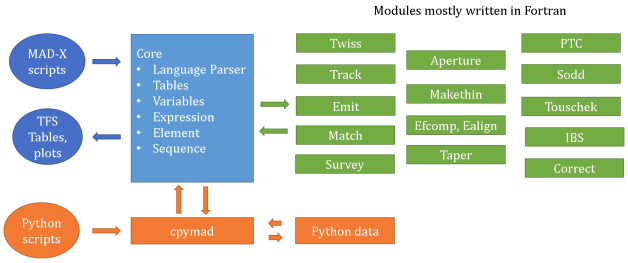

Figure 1: MAD-X Code structure. The core is written in C, with modules mainly in Fortran. cpymad provides a Python interface using the C functions of the core.

CODE STRUCTURE

The MAD-X code is organized as a C core that manages memory and user interface and a set of modules that perform physics computations (see Fig. 1). The user controls MAD-X using a specific scripting language and obtains results in the standard output or tables and plots. The modules implementing the physics models are written in Fortran. The most used module is Twiss which computes optics functions using the transport matrix formalism up to the second order. The module is optimized for speed to be used in optics optimization problems, however, it is accurate only to first order in the closed orbit.

NEW FEATURES

MAD-X has been proven to be a very solid platform for the LHC and its injectors. New machines in the study phase pose new challenges [4]. FCC-ee requires a robust synchrotron radiation effects implementation and more accurate optics calculation due to the large integer tune. Both FCC-ee and EIC require robust calculation around closed orbit far from the origin and magnetic axis deviating from the beam reference paths. The following additional features were developed to support those use cases.

Exact Option

The MAD-X TWISS module, like in MAD8 [6], uses a set of scaled coordinates of a single particle to ensure that they have small values. The positions and momenta of a particle in the accelerator are defined by the coordinate $\vec{z} = (x, p_x, y, p_y, t, p_t)$ as a function of the parameter s. They are related to the global coordinates by the following equation

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-WEPL101

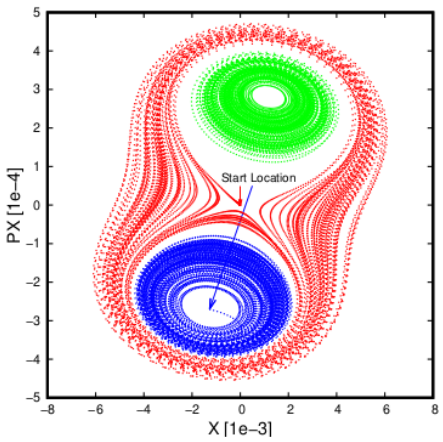

Figure 2: Example for PS single particle tracking with space charge (SC) using the adaptive model [5]. It demonstrates a second order SC resonance in the vicinity of the horizontal integer tune. Three particles are started in each of the two islands and one in the chaotic region around the stable islands.

$$\vec{Q}(s) = \vec{R}(s) + x(s)\hat{X}(s) + y(s)\hat{Y}(s) \quad (1)$$

$$t(s) = \frac{1+\eta\delta_s}{\beta_s}s - cT(s)\,, \quad (2)$$

where $\vec{Q}(s)$ is the position at time $T(s)$ of the particle in the co-moving reference frame positioned on the reference trajectory $\vec{R}(s)$ with the orthogonal axis $\hat{X}(s), \hat{Y}(s), \hat{S}(s) = R'(s)$ and the canonical momenta

$$p_x = \frac{P_x}{P_s} \qquad p_y = \frac{P_x}{P_s} \qquad p_t = \frac{E - E_s}{cP_s}, \quad (3)$$

where δ_s defines the momentum deviation of the reference momentum $P_s = P_0(1+\delta_s) = m\beta_s\gamma_s = \beta_s E_s$ with respect to design momentum P_0 and η is a parameter that is adjusted to keep the design revolution period. As the fields are scaled by P_0, δ_s effectively controls the radial steering, and it is used to introduce a momentum deviation of the machine with respect to the reference momentum. In these coordinates, the Hamiltonian is:

$$H = \frac{1+\eta\delta_s}{\beta_s}p_t - (1+hx)\left(\frac{qA_s}{p_s} + \sqrt{(1+\delta)^2 - \left(p_x - \frac{qA_x}{P_s}\right)^2 - \left(p_y - \frac{qA_y}{P_s}\right)^2}\right), \quad (4)$$

where $1+\delta = \sqrt{p_t^2 + 2p_t/\beta_s + 1}$, $A_{x,y,s}$ is the vector potential, h the curvature of the reference trajectory and q the particle charge.

MAD-X TWISS computes linear transfer matrices $R_{ij}(z) = \frac{\partial M_i(\vec{z})}{\partial z_j}$ from the transfer map $\vec{M}(\vec{z})$ using the approximation $R_{ij}(\vec{z}) = R_{ij}(\vec{0}) + 2\sum T_{ijk}(\vec{0})z_k$, with $T_{ijk} = \frac{1}{2}\frac{\partial^2 M_i(\vec{z})}{\partial z_j z_k}$. This approximation is valid when $\vec{z} \sim \vec{0}$, which is sufficient in many use cases thanks to the coordinate choice and the extra parameter δ_s. However, the approximation is too coarse for those cases, such as the FCC-ee with synchrotron radiation, for which $\delta(s)$ varies considerably in the machine, and EIC that needs the particle to be far from the reference trajectory.

For this reason, the `exact` option has been introduced in TWISS to compute $R_{ij}(\vec{z})$ using exact expressions for the drift. In the new release, new elements are now supported such as translations, rotations, and solenoids, which are solutions of following Hamiltonians

$$H_{\text{rotx}} = +y\sqrt{(1+\delta)^2 - p_x^2 - p_y^2} \quad (5)$$

$$H_{\text{roty}} = -x\sqrt{(1+\delta)^2 - p_x^2 - p_y^2}, \quad (6)$$

and

$$H_{\text{sol}} = \frac{1+\eta\delta_s}{\beta_s}p_t + - \sqrt{(1+\delta)^2 - \left(p_x + \frac{qB_s y}{2P_s}\right)^2 - \left(p_y + \frac{qB_s x}{2P_s}\right)^2}, \quad (7)$$

respectively [7, 8]. Results have been benchmarked with PTC [9], showing exact agreement. An exact formalism for quadrupoles and dipoles is being developed [4].

Space Charge

The space charge (SC) module [10, 11] is integrated in the main branch and therefore available to all users. It has been used in a study for the PS. Figure 2 shows a peculiar example where close to the horizontal integer tune due to the influence of the SC force a second order resonance is exited.

IMPROVEMENTS

Tapering

MAD-X allows correcting magnet strengths to adapt them to the energy of the closed orbit in lattices with synchrotron radiation. In the new version, MAD-X has introduced the parameter `ktap` in dipoles, quadrupoles, sextupoles, octupoles, and multipole elements which contain the tapering compensation, that is numerically computed during the closed orbit search. Tests are ongoing to validate the new functionality on FCC lattices [12].

MADX-PTC

During the validation of the new code MAD-NG [13, 14], many comparisons between MADX-PTC and MAD-NG were made, which led to many improvements in the former. A few simple fixes have been performed in PTC: the multipole fringe field was applied *twice* in some cases, the `old_thick_bend` was reinstated because the new SBEND map [15] was giving imprecise results for small angles in the CLIC BDS study [16], and `n_wedge` was set to zero to avoid

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-WEPL101

many needless computations. The TPSA [17] output format was improved to increase its precision for accurate comparison. Other improvements concern MAD-X to access features already present in PTC and MAD-NG: integration method 8 recently added to PTC, fringe fields for magnets other than SBEND with many previously ignored attributes, namely `h1`, `h2`, `fint`, `fintx`, `hgap`, `f1`, `f2` and `fringe_max` (see MAD-X user manual for definitions [1]).

A detailed comparison of RDTs calculated by MADX-PTC with other codes, including MAD-NG, SUSSIX, and SODD, revealed a few incorrect results which have been fixed [18–20].

The ability of PTC to track the spin of the particle was activated from MAD-X and the results are delivered in a TFS table. The code is being benchmarked for FCC-ee studies showing close results with BMAD, but a few discrepancies still need to be understood, see Fig. 3.

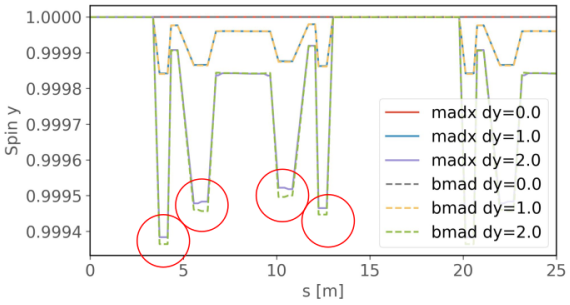

Figure 3: Comparison between BMAD [21] and MADX-PTC spin calculations for FCC-ee [22] as function of the vertical closed orbit.

INTEROPERABILITY

It becomes increasingly important to integrate software in common scientific frameworks to take advantage of the enormous quantity and quality of libraries that are used in optics design (numerical solvers, accelerator control system interfaces, database queries, dashboards, web apps, and GUI, to name a few). For this reason, we decided to support `cpymad` [23] as the official Python bindings for MAD-X. cpymad implements a low-level Python module written in Cython that wraps the internal C data structure of MAD-X and a high-level object that allows the execution of MAD-X commands and provides user-friendly read-write access to the data in MAD-X. The level of introspection surpasses what is exposed to the native MAD-X language and, together with the Python language, it creates a very flexible and powerful environment for scripts and interactive sessions. The high-level interface, by default, instantiates a separate Python process with MAD-X loaded as a shared library. This decouples the main user-facing Python interpreter which allows running multiple MAD-X instances at the same time and does not crash the main Python interpreter in case of fatal error and segmentation fault (unfortunately still easily triggered by some invalid MAD-X usage and

```
from cpymad.madx import Madx

mad=Madx()
mad.call(file="lhc.seq")
mad.input("call,file=optics.madx;")
mad.use(sequence="lhcb1")

for el in mad.sequence.lhcb1:
    if el.name.startswith('mqx'):
        print(el)

for name,value in mad.globals.items():
    if name.startswith('kqx'):
        print(name, value)

tw=mad.twiss()
import matplotlib.pyplot as plt
plot(tw.s,tw.betx)
```

Figure 4: Example of a cpymad script exposing some functionality of the high-level interface and interoperability with other libraries from the Python ecosystem.

bugs). The overhead of passing through the C-API, the first Python interpreter, and the user-facing interpreter are small compared to heavy calculations such as Twiss or Track, but significant for tight loop, sequence constructions, and entire data extraction. There is a potential for improvements, that could be implemented if deemed needed.

The Python interface offers a smooth transition from MAD-X scripting to Python scripting (see Fig. 4), as it is sufficient to call from Python a MAD-X script and take advantage of the more powerful Python interface. It is expected that the community will move slowly to using Python, but there are no plans to discontinue the MAD-X scripting language. At the same time, the MAD-X scripting language, which suffers from some ill-defined grammatical constructs and permissive parses, will not be developed further.

OUTLOOK AND CONCLUSION

MAD-X continues to be used to support beam dynamics studies for existing and future machines. New physics and more accurate models have been recently implemented, motivated by the specific needs of machines such as FCC-ee and EIC. MAD-X is part of the Python scientific ecosystem thanks to the cpymad package that offers very flexible scripting capabilities, extensive introspection, and seamless interoperability with the main Python packages.

Efforts are ongoing to overcome design limitations on the accuracy of TWISS calculations, as well as better supporting integration and interoperability with other software packages.

REFERENCES

[1] *MAD-X project*. `https://cern.ch/madx`

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-WEPL101

[2] A. Abada *et al.*, "FCC-ee: The Lepton Collider: Future Circular Collider Conceptual Design Report Volume 2," *Eur. Phys. J. ST*, vol. 228, no. 2, pp. 261–623, 2019. doi:10.1140/epjst/e2019-900045-4

[3] F. Willeke, "Electron Ion Collider Conceptual Design Report 2021," 2021. doi:10.2172/1765663

[4] G. Simon, A. Faus-Golfe, R. D. Maria, T. Pieloni, and L. V. Riesen-Haupt, *Review of MAD-X for FCC-ee studies*, these proceddings.

[5] F. Schmidt and F. Asvesta, "Space Charge Resonance Analysis at the Integer Tune for the CERN PS," *JACoW HB*, vol. 2021, pp. 124–128, 2022. doi:10.18429/JACoW-HB2021-MOP20

[6] H. Grote and F. C. Iselin, *The MAD program (methodical accelerator design): version 8.10 ; user's reference manual; 3rd ed.* CERN, 1993. https://cds.cern.ch/record/248416

[7] É. Forest, *Beam Dynamics: A New Attitude and Framework*. Hardwood Academic / CRC Press, 1998, vol. 8.

[8] J. S. Berg, "Exact maps to second order for selected elements in mad-x variables," Tech. Rep. BNL-224197-2023-TECH, 2023. doi:10.2172/1968831

[9] F. Schmidt, E. Forest, and E. McIntosh, "Introduction to the polymorphic tracking code: Fibre bundles, polymorphic taylor types and exact tracking," 2002.

[10] V. Kapin and F. Schmidt, "MADX-SC Flag Description," 2013. https://cds.cern.ch/record/1626799

[11] F. Schmidt, Y. Alexahin, A. Latina, and H. Renshall, "3D Symplectic Space Charge Implementation in the Latest Mad-X Version," *JACoW HB*, vol. 2021, pp. 129–134, 2022. doi:10.18429/JACoW-HB2021-MOP21

[12] M. Hofer, *Private communication*.

[13] L. Deniau, *MAD-NG's Reference Manual*. https://cern.ch/mad/releases/madng/html/

[14] L. Deniau, *MAD-NG Source Repository*. https://github.com/MethodicalAcceleratorDesign/MAD

[15] D. Sagan, "Singularity-free Exact Dipole Bend Transport Equations," *JACoW NAPAC*, vol. 2022, 2022. doi:doi:10.18429/JACoW-NAPAC2022-TUPA16

[16] E. Manosperti, L. Deniau, J. Gray, R. Tomás, and A. Pastushenko, "MAD-NG For Final Focus Design," *JACoW IPAC*, vol. 2023, 2023.

[17] L. Deniau and C. Tomoiagă, "Generalised Truncated Power Series Algebra For Fast Particle Accelerator Transport Maps," *JACoW IPAC*, vol. 2015, 2015. https://cds.cern.ch/record/2141771/files/mopje039.pdf

[18] F. Schmidt, *Normal Form tests with tracking data*, BE-ABP/LNO Meeting, 2022. https://indico.cern.ch/event/1224204/contributions/5149953/attachments/2552567/4397701/LNO_23.11.2022-4.pdf

[19] F. Schmidt, "Deriving the Normal Form F Terms with SUSSIX Simulation Analysis," *to be published*, 2023.

[20] L. Deniau, *Methodical Accelerator Design, Update on Normal Forms in MAD-NG*, BE-ABP/LNO Meeting, 2022. https://indico.cern.ch/event/1180836/contributions/4960311/attachments/2479246/4258328/ld_2207-mad_lno_nf_slides.pdf

[21] D. Sagan, *Bmad, a subroutine library for relativistic charged-particle dynamics*. https://www.classe.cornell.edu/bmad

[22] F. Carlier, *Update on xsequence, spin, radiation tracking*, EPFL-LPAP FCC-ee Software Framework Meeting. https://indico.cern.ch/event/1133146/contributions/4754454

[23] T. Gläßle, R. De Maria, Y. Levinsen, and K. Fuchsberger, *cpymad*, version v1.13.0, 2023. doi:10.5281/zenodo.7752353