MAD-X Progress and Future Plans
ICAP 2012, Warnemünde

Laurent Deniau
CERN-BE/ABP

23\textsuperscript{th} August 2012
Part I — Status
Single Particle Beam Dynamic Code
- Motion of particles in 6D phase space under external fields (e.g. Lorenz Forces)
- MAD language (lattice description)
- Modular for the physics

Linear motion (MAD-X)
- 2D geometry (stacked elements + field and alignment errors)
- Element slicing (Makethin, drift-kick-drift)
- 2\textsuperscript{nd} order optics functions (Twiss, $X,R,T$, not symplectic)
- High order thin tracking (Track, $X$, symplectic)
- Optimization (Match)
- Other modules: Survey, Aperture, Correct, Dynap, IBS, Emit, Touschek, Plots, ...

Non-linear motion (PTC)
- 3D geometry
- Differential algebra, Taylor maps
- High order tracking (Lie maps)
- High order analysis (normal forms)
Management philosophy

- One custodian (person centric)
- Modules keepers (collaborations)
- Almost frozen during 2006-2011
- Last big improvements
  - PTC/FPP inclusion into MAD-X (2003)
  - PTC_TWISS, PTC_TRACK, PTC_NORMAL (2005)

Programming philosophy

- Mixed C (Core), C++ (TPSA), Fortran 90 (PTC), Fortran 77 (MAD8)
- ~165K SLOC (50% PTC), ~40K SLOC in C/C++
- MAD-X code is not modular (global namespace & variables)
- PTC is modular (F90 modules)
- Release for Linux (Mac and Windows aside)
Emotional interlude (Oct. 2011)
Improvement strategy

๏ Keep it working
  ➡ Debug and request follow-up
  ➡ Legacy code support
  ➡ Service centric

๏ Improve the existing
  ➡ From outside to inside layers
  ➡ Structured documentation
  ➡ Build & test uniformity (Windows, Mac, Linux)
  ➡ Disentangle I/O (C vs. Fortran I/O)
  ➡ Ensure no regression
  ➡ Save resources

๏ Code reorganization (Preliminaries to new development)
  ➡ Enforce the cohesion (close modules)
  ➡ Reduce the coupling (remove globals)
  ➡ Reduce the complexity (new design)
  ➡ Improve reentrancy (new design)
**Project status** *(roadmap phase-1: 24 tasks)*

**Tasks**

- reorganize source code
- setup build system
- setup website
- setup tracker
- setup test system

**Objectives**

- enforce the cohesion
- fast & portable builds
- information & resources
- bugs & request follow-up
- new tested release

- Restore invariants
- Service centric
- How much testing is enough?
New structure
Easy access to material, information, documents, examples, releases, mailing lists...
Project roadmap

Website: http://cern.ch/mad

---

MAD - Methodical Accelerator Design
CERN - BE/ABP Accelerator Beam Physics Group

Introduction

MAD is a project with a long history, aiming to be at the forefront of computational physics in the field of particle accelerator design and simulation. The MAD scripting language is de facto the standard to describe particle accelerators, simulate beam dynamics and optimize beam optics.

MAD-X is the successor of MAD-8 and was first released in June, 2002. It offers most of the MAD-8 functionalities, with some additions, corrections, and extensions. The most important of these extensions is the Polymorphic Tracking Code (PTC) of E. Forest (see documentation).

MAD-X is released for the Linux, Mac OS X and Windows platforms for 32 bit and 64 bit architectures (see releases). The source code is written in C, C++, Fortran90 and Fortran77. The architecture of MAD-X is under complete review and reorganization in order to improve its maintainability, its flexibility and its performance with full backward compatibility. This long process should be completely transparent for the end users (see roadmap).

The support and maintainance strategy of MAD-X is based on the module keepers/helpers to debug and improve the legacy code (see contributors). This task is extremely complex within the legacy code and finding effective correction can take significant time.

---

MAD on the web

- MAD  http://cern.ch/mad  (this website, currently an alias for madx)
- MAD-X  http://cern.ch/madx
- MAD-9  http://cern.ch/mad9
- MAD-8  http://cern.ch/mad8

The links above are the officially maintained MAD links, but the previous MAD-X website is still online.
## {6} All Tickets By Milestone (Including closed) (173 matches)

A more complex example to show how to make advanced reports.

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>#173</td>
<td>Invalid format does not report an error</td>
</tr>
<tr>
<td>#172</td>
<td>Proposal for tests and changes in the MADX code</td>
</tr>
<tr>
<td>#171</td>
<td>Parser silently ignore **</td>
</tr>
<tr>
<td>#170</td>
<td>Numdiff problem in test-jacobian-knobs</td>
</tr>
<tr>
<td>#169</td>
<td>PLOT crash</td>
</tr>
<tr>
<td>#168</td>
<td>Test match is failing</td>
</tr>
<tr>
<td>#167</td>
<td>Connect beam-beam PTC element to MAD-X element</td>
</tr>
<tr>
<td>#166</td>
<td>Does not work on Windows</td>
</tr>
<tr>
<td>#165</td>
<td>Too many commas in twiss command not handled</td>
</tr>
<tr>
<td>#164</td>
<td>Problem with plot when generating postscript file</td>
</tr>
<tr>
<td>#163</td>
<td>Jacobian run infinitely when madx is compiled with gfortran</td>
</tr>
<tr>
<td>#162</td>
<td>DA becomes unstable during ptc_twiss</td>
</tr>
<tr>
<td>#161</td>
<td>Output is 80 columns when compiled on Windows with Fort</td>
</tr>
<tr>
<td>#160</td>
<td>Output is desynchronized when compiled with gfortran &gt;= 4.4</td>
</tr>
<tr>
<td>#159</td>
<td>Redirection not detected on Windows</td>
</tr>
<tr>
<td>#158</td>
<td>Multiple kick in Track ignores parameter deltap</td>
</tr>
<tr>
<td>#157</td>
<td>Aperture documentation for racetrack shape</td>
</tr>
<tr>
<td>#156</td>
<td>Use madx as a shared library build</td>
</tr>
<tr>
<td>#155</td>
<td>Implicit drifts produced differently</td>
</tr>
<tr>
<td>#154</td>
<td>Improve maketime tepat method beyond 4 slices</td>
</tr>
<tr>
<td>#153</td>
<td>Memory overflow and input bug on windows</td>
</tr>
<tr>
<td>#152</td>
<td>PTC does not find the closed orbit</td>
</tr>
<tr>
<td>#151</td>
<td>Significant memory leak in PTC</td>
</tr>
<tr>
<td>#150</td>
<td>Plot with interpolate changes value of bending angle</td>
</tr>
<tr>
<td>#149</td>
<td>Lost particles differ from previous version</td>
</tr>
<tr>
<td>#148</td>
<td>Incorrect beam parameters retrieval</td>
</tr>
<tr>
<td>#147</td>
<td>Silent bug in the parser when ** is forgotten.</td>
</tr>
<tr>
<td>#146</td>
<td>Beam parameters are set to electron</td>
</tr>
<tr>
<td>#145</td>
<td>Buseror in PTC twiss</td>
</tr>
<tr>
<td>#144</td>
<td>Invalid dispersion calculation with expression for e1, e2</td>
</tr>
</tbody>
</table>

Results (1 - 100 of 173)
Principle: *compare the outputs with the references under constraints*

Ensure consistent results through cross-platforms

---

**Test system: black-box testing**
Test system: test levels

- **Unit tests** (not implemented)
  - very fast, should take <10 seconds
  - run as often as possible
  - requires a complete redesign of madx code (not foreseen in 2012)

- **Test cases**
  - fast, should take <10 minutes
  - light, should not depend on large lattice files
  - run during development to validate **modules** after changes in the code

- **User cases**
  - longer, can take hours
  - can depend on large lattice files shared across studies (e.g. LHC optics)
  - run before releasing to validate **studies** and avoid backward incompatibilities
How numbers are compared?

- **Absolute:** \(| a - b | \leq \text{abs}\)

- **Relative:** \(| a - b | \leq \text{rel} \cdot \min(| a |, | b |)\)

- **Digits:** \(| a - b | \leq \text{dig} \cdot \min(| a |, | b |) \cdot 10^{-\text{ndig}}\)

  - \(\text{ndig}\) is the max number of significant digits read from input (i.e. adaptive)

- \(a = 0\) or \(b = 0\) \(\Rightarrow\) \(\text{min} = 1\)

  - Relative \(\Rightarrow\) Absolute

What is a line? *(row count)*

- Everything up to `\n` (Unixes), `\r\n` (Windows) or `\r` (Old Mac)

  - Portable across operating system

Constraints can overlap

- Last constraints in the constraints file prevail

- Define weak constraints first, then refine tolerances

- Rule #0 (default) is equivalent to “* * abs=DBL_MIN” (DBL_MIN = 2.22507e-308)
Test system: examples

- Configure tests
  # Test config for the Jacobian knobs (test-jacobian-knobs.cfg)
  # rows  cols  constraints
  1-7    *    skip  # head banner
  149-$  *    skip  # tail banner
  # first matching
  37-38  1-2  rel=1e-12
  39     2    abs=1e-21  # from job
  41     1    rel=1e-12
  # second matching
  109-110 1-2  rel=1e-12
  111    2    abs=1e-21  # from job
  113    1    rel=1e-12

- Run tests

  [ Jacobian testsuite ]
  + test-jacobian   (0.00 s) - 1/ 1 : PASSED
  + test-jacobian-2 (0.00 s) - 1/ 1 : PASSED
  + test-jacobian-knobs (0.00 s) - 2/ 2 : PASSED

  [ RF multipole testsuite ]
  + test-rfmultipole (0.00 s) - 9/ 9 : PASSED
  + test-rfmultipole-2 (0.00 s) - 2/ 2 : PASSED
  + test-rfmultipole-3 (0.00 s) - 2/ 2 : PASSED
  + test-rfmultipole-4 (0.00 s) - 2/ 2 : PASSED

  [ PTC Twiss testsuite ]
  + test-ptc-twiss   (0.00 s) - 4/ 4 : PASSED
- New structure
- Better language description
- Clarify lattice description
- Improve the learning curve of new comers

---

### MAD Language

#### Keywords

- **Identifiers**
  
  An identifier is a sequence of letters, digits, decimal points, and/or underscores that the user chooses. An identifier has to begin with a letter; it cannot begin with a digit. Characters beyond the sixteenth are dropped. Uppercase and lowercase letters are treated indistinctly as lowercase (i.e. case insensitive).

  **Syntax:**

  \[ [a-zA-Z][a-zA-Z0-9_] \]

  **Examples:**

  - `abcd`, `ab_cd`, `a00b10`, `ab.cd` (valid identifiers)
  - `10abcd`, `_abcd`, `.abcd`, `ab1cd` (non-valid identifiers)

- **Labels**

  A label is made up of an identifier followed by `:`.

  **Syntax:**

  identifier-name:

- **Reserved Keywords**

  MAD keywords are reserved identifiers, using MAD keywords as identifiers result in an error.
Emotional interlude (May 2012)
New element: RF-Multipole

๏ Thin multipole element with RF modulation (rotating harmonics) and longitudinal kick (acceleration)

๏ Harmonics: \[ B_n + iA_n = \frac{1}{n!} [K_{N,n}L + iK_{S,n}L] = \frac{1}{n!} [K_{N,L,n} + iK_{S,L,n}] \]

๏ Hamiltonian: \[
H = -\frac{1}{k_{RF}} \frac{qV_{RF}}{p_{sc}} \cos(\varphi_{RF} - k_{RF}z) + \sum_{n=0}^{N} \frac{1}{(n+1)!} \text{Re} \left[ \left( K_{N,L,n} \cos(\varphi_{n} - k_{RF}z) + iK_{S,L,n} \cos(\varphi_{n} - k_{RF}z) \right) (x + iy)^{n+1} \right]
\]

๏ Effects relative to the reference particle: (a) multipole, (b) RF-multipole

![Diagram](attachment://image.png)

**Courtesy to**

R. De Maria (Hamiltonian)
A. Latina (implementation)
New element: Nonlinear elliptical lens

- Nonlinear thin lens with potential of elliptic shape

\[ U(x, y) = \frac{k \xi \sqrt{\xi^2 - 1} \text{acosh} \xi + \eta \sqrt{1 - \eta^2 (\cos \eta - \pi/2)}}{\xi^2 - \eta^2} \]

\[ \xi = \frac{\sqrt{(x + c)^2 + y^2} + \sqrt{(x - c)^2 + y^2}}{2c} \]

\[ \eta = \frac{\sqrt{(x + c)^2 + y^2} - \sqrt{(x - c)^2 + y^2}}{2c} \]

- KNLL: The integrated strength of lens [m]. The strength is parametrized so that the quadrupole term of the multipole expansion is \( k1 = 2 \times \text{KNLL}/\text{CNLL}^2 \)

- CNLL: The dimensional parameter of lens [m]. The singularities of the potential are located at \( X = -\text{CNLL}, +\text{CNLL} \) and \( Y = 0 \)

Courtesy to A. Valishev (implementation)
Part II — Future
Purpose of optics codes

1. **Definition**: define or modify machine parameters using the MAD language.

2. **Tracking**: track particles or maps to find periodic, quasi-periodic or constrained solutions, i.e. one-turn map and closed orbit.

3. **Analysis**: compute optics functions for the one-turn map, use normal forms for high-order terms.

4. **Optimization**: optimize the design with user-defined constraints, e.g. interaction regions matching.

5. **Validation**: perform single-particle tracking campaign to validate the design, e.g. check the dynamic aperture.

---

The amazing discrepancy between codes has its origin mainly in the integrator scheme

L. Nadolski
Optic calculations

- **Lattice model**
  - Track: D-K-D, symplectic, 0\textsuperscript{th} order maps (orbit)
  - Twiss: M-K-M, non-symplectic, 2\textsuperscript{nd} order maps (orbit, R, T)
  - PTC: D-K-D & M-K-M, symplectic, exact, high order maps (TPSA)

- **Symplectic integration (?)**
  - Knowledge of the energies (Hamiltonian)
  - Knowledge of the transfer maps (motion)
  - Knowledge of the slicing scheme (order)

- **Tracking maps**
  - Differential algebra, Taylor maps
  - Truncated power series algebra (TPSA)

- **One-turn-map analysis (?)**
  - Normal forms (Jordan forms)
  - Lie algebra (non-linear analysis)

- **Matching** (optimization, fixed points)
- **Dynamic aperture** (long-term behavior)
Symplectic integrators: approximations

- Lattice models
- Element models
- Integrators schemes
  - Approximate solution to exact Hamiltonian
  - Exact solution of approximate Hamiltonian
- Approximations
  - composition method
  - series truncation
  - paraxial approximation
  - radius of convergence
  - Lorenz transform (0 < $\beta$ ≤ 1)
- Parameters
  - small $p_0 c$
  - large $\Delta p/p$
  - fixed point stability (beamlines closed orbit)
  - large aperture $A$ vs. integrated strength $KL$
  - off momentum, off axis beams vs. small field length $L$ and curvature radius $\rho$
Emotional interlude (now)
Symplectic integrators: benchmarking

- Element models
  - O. Berrig, “Comparison of transfer maps of PTC and MAD-X for the dipoles magnets”, CERN, 2008

- Lattice models
  - E. Keil, “Emma in MAD-X and comparison with other programs”, CERN ATS note 2010-044 (FFAG)
  - M. Giovannozzi, “Multi-turn extraction studies and PTC”, CERN, 2011.

- Symplectic integrators models
Simple Cyclotron (SBENDs)

![Graph](image_url)

- PTC exact
- MAD-X 5 SBENDs
- MAD-X 50 SBENDs

Courtesy to F. Schmidt
Simple Cyclotron (SBENDs)
Symplectic integrators: high order

- Simple Cyclotron (SBENDs)

![Graph](image-url)
Symplectic integrators: application

- Multi-turns extraction in the CERN-PS: MAD-X/PTC simulations

![Graph showing multi-turns extraction](image)

**End of adiabatic capture (1/4 resonance)**

- **y [mm]**
  - **PS internal**
  - **PS external**

- **x [mm]**

**ΔQ vs Δp/p**

- **Delta Q H - island**
- **Delta Q V - island**
- **Delta Q H - core**
- **Delta Q V - core**

**SS01, qx=0.261, qy=0.30**

*Courtesy to M. Giovannozzi*
Application logic: complaints

- Classification of complaints
  - 77% core, 23% physics

- Major problems
  - Memory leaks (one shot run)
  - Data sharing (no ownership)
  - Data lookup (dangling pointers, string comparison, performance)
  - Side effects (no reentrancy, interdependence)

<table>
<thead>
<tr>
<th>Groups</th>
<th>#</th>
<th>Tickets no</th>
</tr>
</thead>
<tbody>
<tr>
<td>interpreter</td>
<td>14</td>
<td>56, 57, 64, 67, 73, 89, 109, 110, 124, 125, 132, 136, 147, 165, 171</td>
</tr>
<tr>
<td>table, select</td>
<td>10</td>
<td>48, 81, 85, 99, 122, 123, 130, 139, 141, 148</td>
</tr>
<tr>
<td>sequence, use</td>
<td>7</td>
<td>61, 74, 80, 89, 93, 120, 126</td>
</tr>
<tr>
<td>plots</td>
<td>12</td>
<td>42, 69, 71, 76, 85, 86, 102, 115, 116, 131, 150, 164</td>
</tr>
<tr>
<td>memory leaks</td>
<td>7</td>
<td>1, 3, 4, 92, 111, 151, 153</td>
</tr>
<tr>
<td>PTC physics</td>
<td>2</td>
<td>75, 118</td>
</tr>
</tbody>
</table>
Application logic: concepts

Concepts
- ownership policy
- runtime polymorphism
- scope & context
- notification policy

Improvements
- memory leaks
- side effects
- reentrancy
- consistency
- functional
- performances
- flexibility

OpenMP
Application logic: interpreter

- Recursive descent parser
  - polymorphic

- Parser requires
  - objects factory
  - ownership policy
  - runtime polymorphism (i.e. interface oriented programming)

- Parser provides
  - syntax, grammar, scopes & contexts
  - very efficient evaluator (e.g. evaluates $x=x+1$, $1.6 \times 10^8$ times per second)

```
stdin → lines → AST → code → eval → effects
```

```
istream → lexer → symbols → parser → contexts
```

```
compile
```

```
parser
```

```
parser
```

```
parser
```

```
parser
```
- Matrix expression $M = M_1 \ast M_2 + M_3 \ast M_4$ evaluated $10^8$ times (dynamic sizes)
- Take advantage of data structure topology vs. intrinsics (beyond compiler optimization)
- Applicable to dense multivariate polynomials (TPSA)
Application logic: documentation

๏ First impression of the project (with the website)
  ➡ Professionalism

๏ Structured documentation
  ➡ Separation of Concerns principle
  ➡ Reflects the code (quality)
  ➡ High priority

๏ On the model of MAD8 and SixTrack
  ➡ MAD-X user’s guide
  ➡ MAD-X physics guide
  ➡ MAD-X developer’s guide
Future plans

- **Short term (2012)**
  - Setup more tests (~100)
  - Cleanup examples
  - Cleanup documentation
  - Debug, test, debug, test, debug, ...
  - Produce pro release 5.01.00

- **Mid-Term (2013)**
  - Benchmarking the physics
  - Physics guide (MAD8-like)
  - Rewrite the core (C part, ~30K SLOC)
  - Cleanup modules (Makethin, Aperture, Survey, Correct)
  - 3D geometry, better use of PTC
  - Plotting system, math kernel

- **Long-Term (2014+)**
  - Clarify models (Hamiltonians, slicing models)
  - Unify modules (Track, Twiss, Makethin, PTC)
  - Improve optimization (Match)
  - Convert/Rewrite Fortran to C (?)
Thank you for your attention

http://cern.ch/mad
mad@cern.ch