Optics Code MAD-X and Tracking code SixTrack at CERN - An Overview

Optics Code MAD-X

Help: L. Deniau

- Why MAD-X?
- MAD-X Strategy
- Project Style – Module Keepers
- 10 Real Treats of MAD-X
- Key Update – PTC
- Limitations
- New Beginning!

Why MAD8 to MAD-X

- MAD8: well debugged but:
  - Program Structure has grown much too complex over the years
  - Zebra Memory Management outdated and maintenance questionable
  - Modules interleaved and not independent
  - New modules hard to integrate
  - Upgrade to TPSA, Normal Form etc practically impossible
- MAD9: modern, C++, object-oriented but:
  - Very complex & slow
  - Our group invested a lot of time to get to run it for the LHC, but to no avail…
  - Hardly any expertise in C++ in our group

➔ In 2000 new Project based on MAD8 called MAD-X to quickly have a tool ready for LHC Design work: 2001 F. Schmidt first custodian

MAD-X Strategy

- No Restrictions of the Programming Language, basically C, Fortran77 & Fortran90
- Core in C: Memory Management, Input Parser & Interfaces
- Truly Independent Modules provided by anyone
- Flexible “open” structure
- CVS version control (later SVN)
- Skip all ambiguous or limited modules
- Concentrate on LHC issues and leave other machine issue for later addition
- Replace Lie Algebra tools by much more powerful TPSA techniques using PTC

Project Style

1. One "custodian" to look after the program core and to oversee module development
2. Team of module keepers
3. Disadvantages: more work for users (Code, Example & Documentation!!!), more disciplined and structured work (CVS etc), less homogeneous code, some modules may be delayed, succession issue
4. Advantages: better sharing of work load, potentially better code since user defined, user steer progress of code, users relate better to code, I.e. improved user satisfaction

Module Keepers

1. Alexander Bolshakov (ITEP) – various additions
2. André Verdier – survey, thintrack
3. Daniel Brandt– ibs
4. Etienne Forest (KEK) – PTC proper
5. Frank Schmidt – Custodian, c6t, twiss, PTC modules
6. Frank Zimmermann – dynap
7. Fulvia Pilat (BNL) – SXF
8. Helmut Burkhardt – makethin, MAC system 10
9. John Jowett – Windows Version
10. Oliver Brüning – match
11. Ralph Aßmann – emit
12. Thys Risselada – threader & MMK secretary
13. Tommaso d'Amico – plot
14. Werner Herr – error, cororbit

10 Real Treats of MAD-X

1. MAD-X **Parser** pretty powerful (albeit not perfect)
2. Sequence editing versatile ➔ example
3. Matching is very versatile, pattern matching etc
4. In-code plotting ➔ famous plots (machine structure on top)
5. All Data in standard TFS tables
6. Macros are a powerful tool to expand on what the code can do ➔ example. The price is speed.
7. Symplectic tracking
8. Documentation (web page)
9. Examples
10. Expert Support

Example I:

Flat and Minimal Lattice

```
! all set-up ready for clean-up
  use, period=MACHINE;
! step 1: super flat real sequence even when started with lines
! "SAVE" and "CALL" needed
  save,sequence=MACHINE,file=blah;
  call,file=blah;
! Now seqedit
  seqedit, sequence=MACHINE;
! All classes that can go
  select,  flag=seqedit, class=marker      ;
  select,  flag=seqedit, class=monitor      ;
  select,  flag=seqedit, class=instrument   ;
  select,  flag=seqedit, class=solenoid     ;
  select,  flag=seqedit, class=kicker       ;
  select,  flag=seqedit, class=multipole,pattern="^eld$" ;
remove,  element=selected;
endedit;
! now use the super clean lattice
use, period=MACHINE;
```

Example II: Matching with PTC

- Using the “match, use_macro” construct as for TWISS

- Example Twiss

```
match, use_macro;
vary, name=kqf;
M1: MACRO {twiss, sequence=lhcb1;dq=table(summ,Q1)-table(summ,Q2)-5;}
constraint, EXPR=table(summ,Q1)-table(summ,Q2)=5;
simplex, tolerance=0.00000001;
endmatch;
```

- Example PTC_TWISS

```
match, use_macro;
vary, name=kqf;
M1: MACRO {ptc_create_universe;
    ptc_create_layout,model=2,method=2,nst=1,exact,resplit,thin=.0005;
ptc_align;
ptc_twiss,closed_orbit, icase=4;ptc_end;}
constraint, EXPR=table(ptc_twiss_summary, Q1,1)-table(ptc_twiss_summary, Q2,1)=5;
simplex, tolerance=0.00000001;
endmatch;
```

Key Features: MAD-X Upgrade PTC 1/2

The code PTC is composed of two distinct parts.

A) The independent polymorphic library FPP* 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.

*(based in an idea by J. Bengtsson and relying on DA packages of M. Berz or others)

Key Features: MAD-X Upgrade PTC 2/2

- PTC is a physics engine only! Offers hardly any tool of convenience, except when absolutely mandatory like closed orbit searcher.
- PTC remains E. Forest's independent (Fortran90) program linked to MAD-X via a convertor.
- 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.
- Design Goal: 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.

Limitations (2009)

- The MAD-X parser in C is by now very complex and difficult to modify.
- The TWISS module (maybe others) have reached their end-of-life status. It exists for performance reasons only.
- The PLOT module is also difficult to maintain. The interactive mode requires libX11.a for a statically linked executable (requires special installation under Linux).
- Documentation is pretty old-fashioned and should be replaced by a professioal 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 to overcome this problem!

Methodical Accelerator Design

- **MAD** project
  - ➡ Website: http://cern.ch/mad *(Services centric).*
  - ➡ Support: mad@cern.ch *(MAD team)*, mad-usr@cern.ch *(MAD community).*
  - ➡ New build & test system (regression tests) for Windows, Linux, MacOSX, 32 & 64 bit.
  - ➡ Garbage collector (memory management), ~10 requests, ~100 bugs corrected (SVN Trac*ker*).
  - ➡ Production release in February 2013 (next pro: July 2013), ~5-7 development releases/year
- MAD team
  - ➡ Laurent Deniau *(project manager)*
  - ➡ Ghislain Roy
  - ➡ Andrea Latina
  - ➡ Piotr Skowronski

- **MAD** (new development)
  - ➡ Focus on PTC/FPP approach, collaboration with KEK (E. Forest), June 2013 ➟ June 2017.
  - ➡ New PTC/FPP engine, better structured, unified & extended physics *(invariants, envelopes)*.
  - ➡ General purpose scripting language, provide access to physics and maths objects *(toolboxes)*.

MAD-X evolution

MAD-X

New development
"the big picture"

general purpose
scripting language

MAD-X businesstracking + analysis + optimization + visualization + modules

provide deep access from the scripting language to
lattice, elements, geometry, maps, normal forms, invariants, integrals, plots

• TT	• CLIC	• LHC
• PS	• ELENA	• HL-LHC
• PSB	• *EMMA*	• HE-LHC
• SPS	• *MedAustron*	• LHeC
• LEIR		•TLEP
• ISOLDE		• MTE
		• *Space charge*

- **Concentrate on PTC/FPP approach**
  "do it right, make it simple, make it fast",
- **Give to users deep access to mathematical and lattice objects**
- **Improve the flexibility and the performance: new TPSA & FPP & PTC**
- **Self-checking physics, easier to use,**
  bug free, x-checked with the existing
- **General purpose scripting language**
  compatible with MAD-X scripts

unified usage & models
avoid backdoors

Tracking Code SixTrack

Help: R. De Maria, E. McIntosh

- Historical Overview
- Dynamic Aperture
- Stable/Chaos/Loss in 1D & 2D
- Survival Plot
- Tracking Engine
- Future Developments

Sixtrack Post-Processing

Historical Overview I

1985-2012 F. Schmidt Era

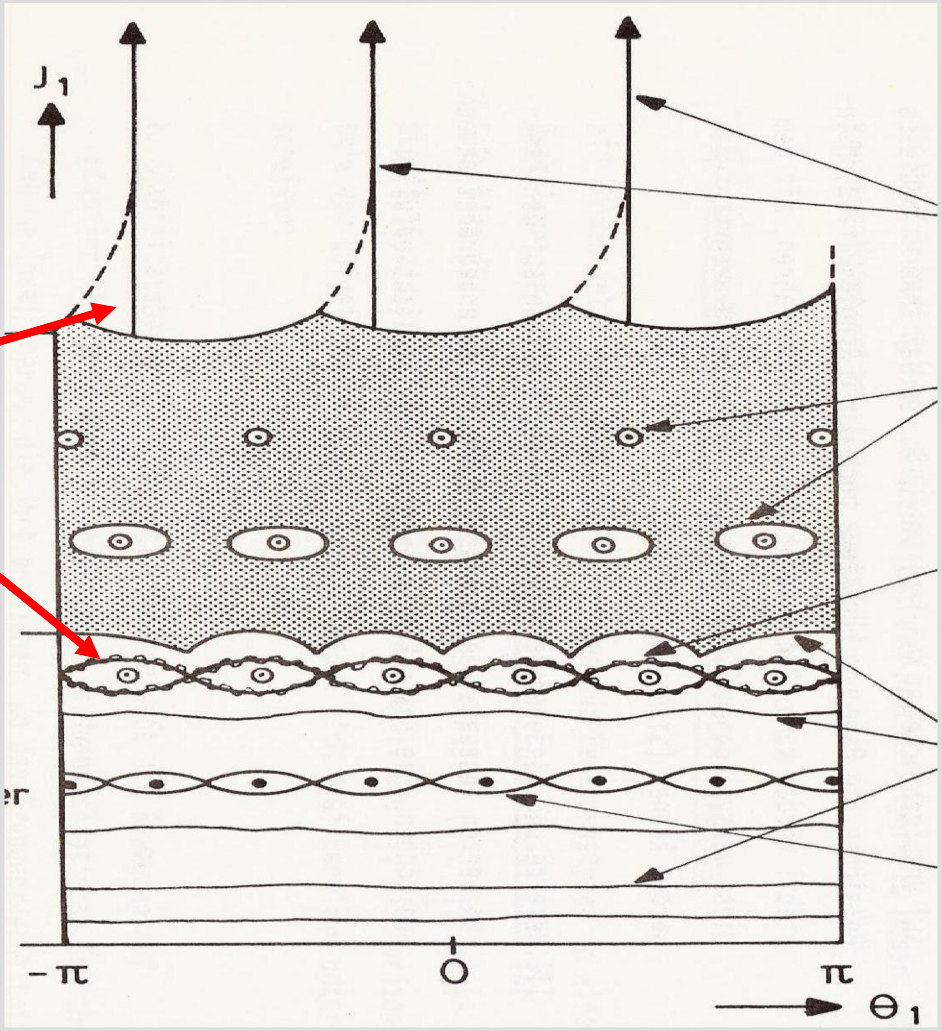
- Racetrack Code early `80 by A. Wrulich, manual 1984
- Symplectic Hamiltonian treatment (1985-2001 by late G. Ripken):
  - 6D with arbitrary energy thick lens
  - Thin Lens approximation
  - 6D coupled beam-beam
- SixTrack started as code in 1990 (agreed with A. Wrulich)
- Sixtrack Developments
  - Manual
  - Astuce code development (H. Grote)
  - CVS later SVN
  - Post-processing for tracking data
  - Differential Algebra, Maps and NormalForm (M.Berz/E. Forest/F. Schmidt)
  - Sixtrack Run Environment
  - MAD-X SixTrack convertor (H. Grote/F. Schmidt)
  - Dynamic Aperture Studies for the LHC

Historical Overview II

E. McIntosh/F. Schmidt 1990-2012

- **SixTrack Run environment professional Development**
- **Speed optimization on: Cray, DEC multicore, PC farms**
- **Vectorization for Cray**
- **Checkpoint-Restart**
- **In-house screen saver CPSS**
- **BOINC up to 200’000 subscribers & 75’000 continuously for 6 months**
- **Bit-by-bit identical results on any IEEE machine (crlibm function library)**
- **Proper bracketing (H. Renshall) ➔ compiler independent**

Dynamic Aperture Scheme



DA???

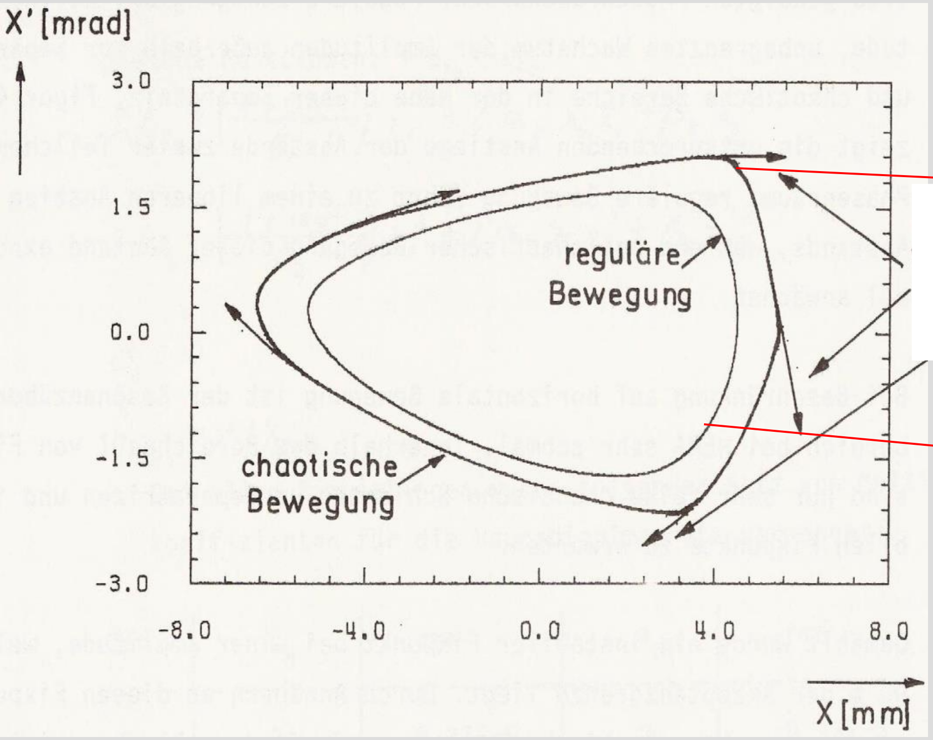
Rapid amplitude growth and loss

Stable Islands in chaotic sea

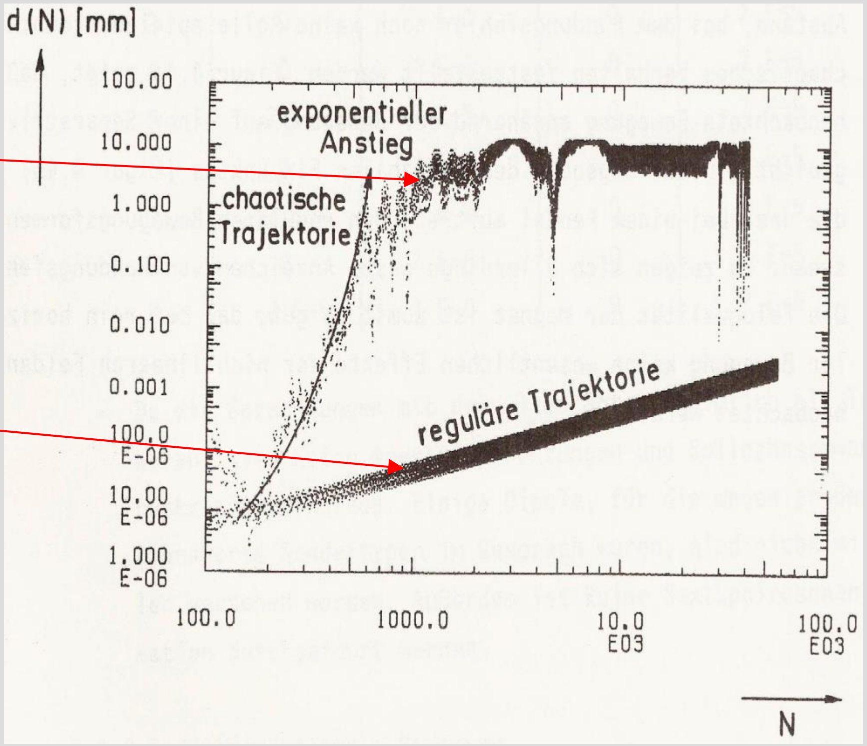
Fine chaotic layers in stable regime

Mostly stable particle motion

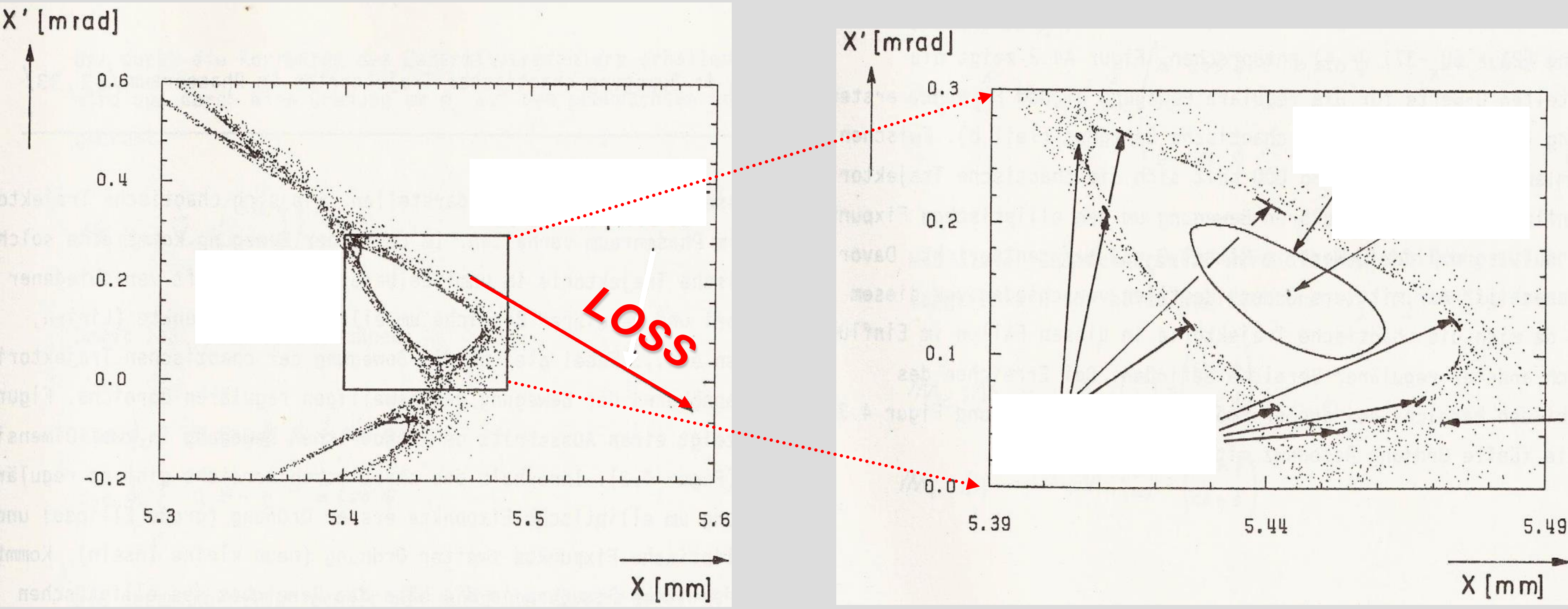
DA in 1D



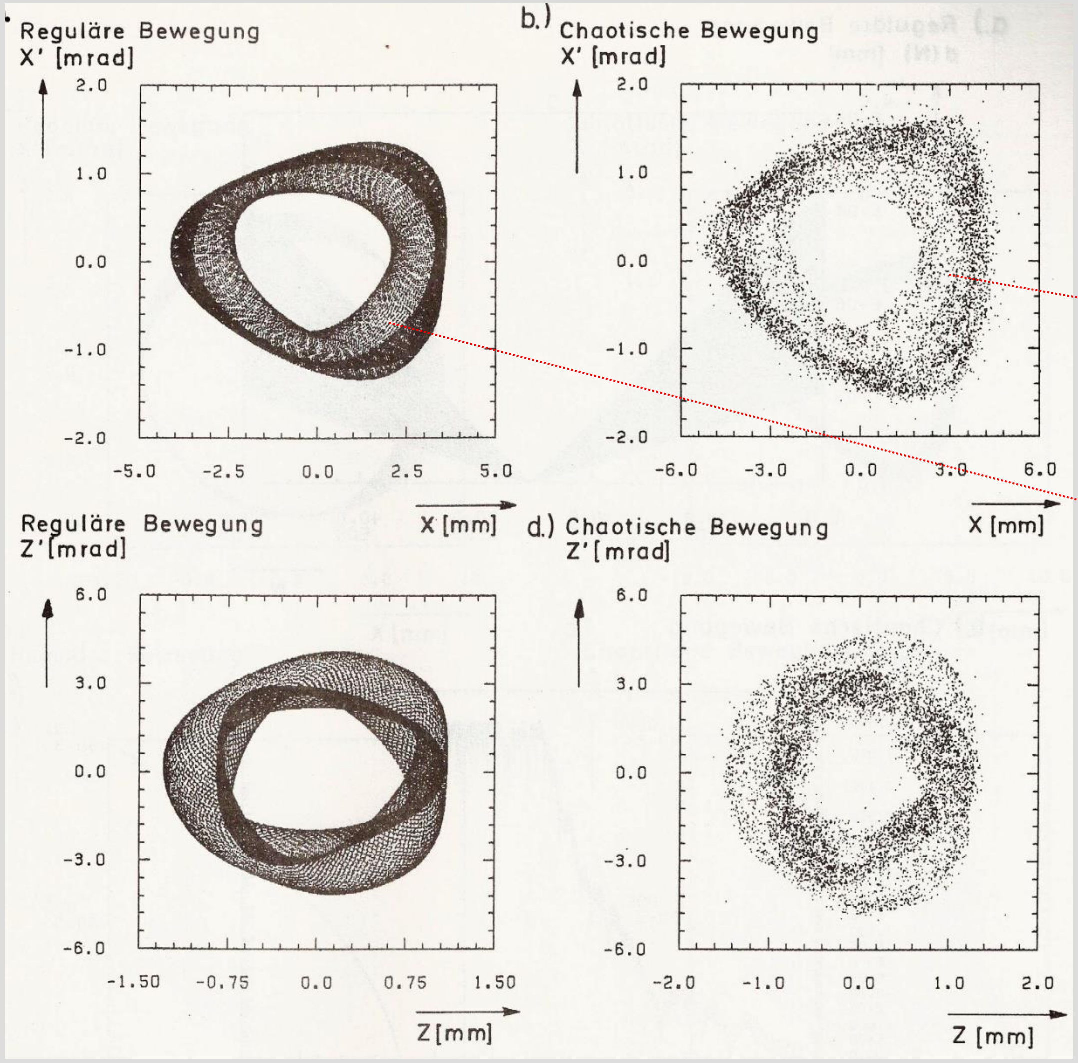
Particle Loss

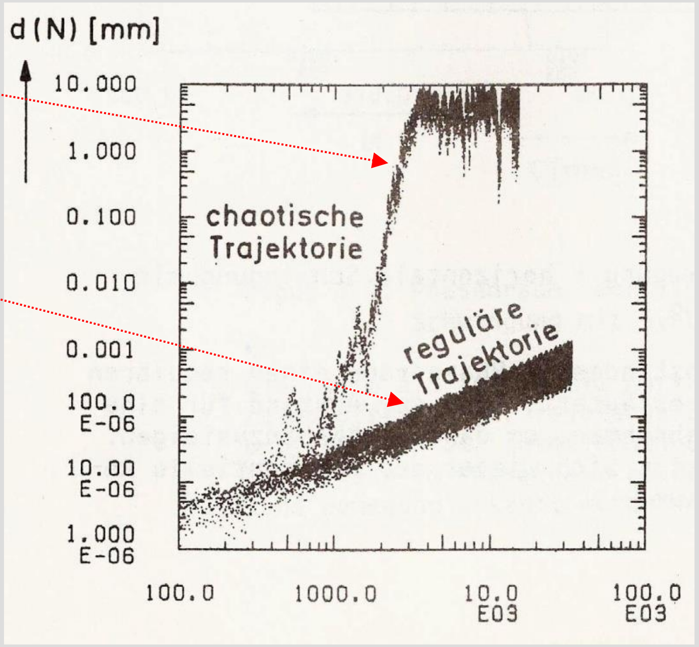


1D close-up

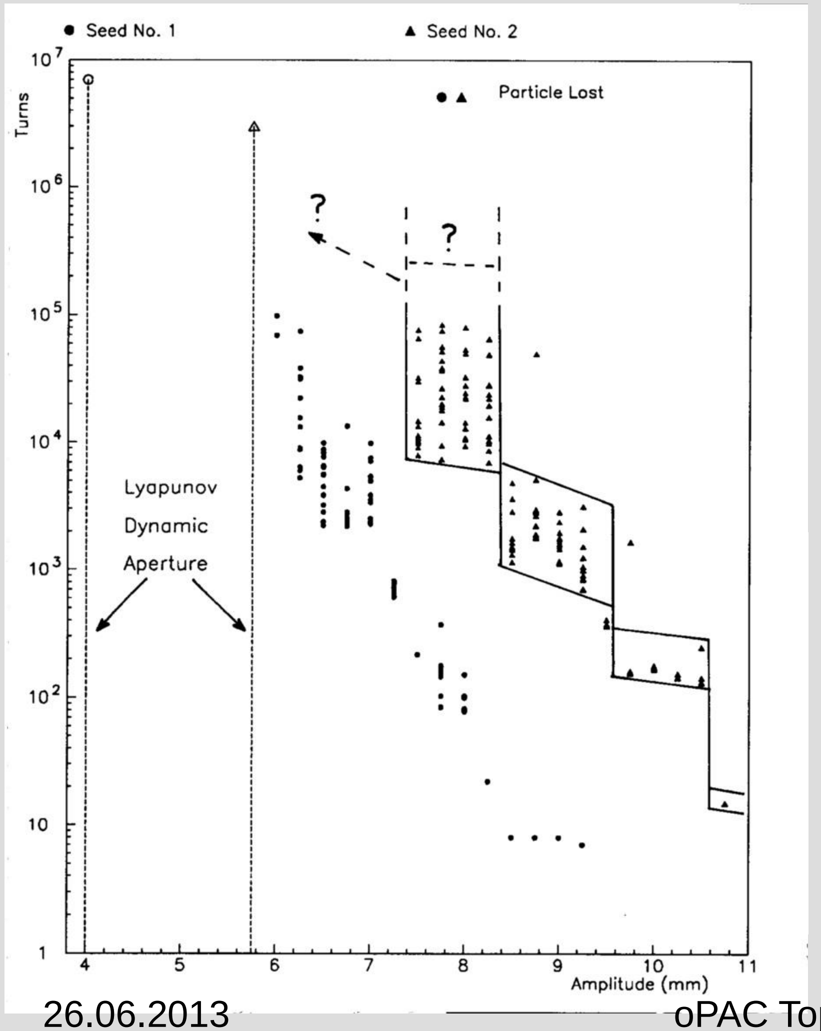

X' [mrad]
0.6
0.4
0.2
0.0
-0.2
5.3
5.4
5.5
5.6
X [mm]
LOSS
X' [mrad]
0.3
0.2
0.1
0.0
5.39
5.44
5.49
X [mm]

2D Stable and Chaos





Survival Plots



The survival plots depict the number of turns particles stay in the machine for a given amplitude. In this particular example we tracked an ensemble of particles started in a tiny volume of phase space. The width of the survival times tends to grow inverse proportionally with amplitude since the chaotic motion becomes weaker. There is no known method that could predict the evolution to smaller amplitudes toward the onset of global chaos.

Tracking Engine for massive LHC Tracking Studies

- For LHC tracking we are using SixTrack which is kept rigorously bug-free while continuously optimizing the code for speed.
- The lattice is transferred from MAD-X. MAD-X, SixTrack & PTC are benchmarked against each other.
- Great care has been taken to sample the 6D phase space appropriately!
- We have prepared a SixTrack run environment which allows to automatically launch 10’000 of jobs at a time and storing all output data in a elaborate directory structure. After all jobs have been finished automatic post-processing procedure are being launched for a full analysis of the data.
- Jobs can be sent to various batch systems, e.g. a local CERN batch cluster with hundreds nodes is at our disposal.
- Moreover, we have created LHC@Home. Presently about 100’000 volunteers have sign up to contribute world-wide. The so-called BOINC system organizes the flow of jobs to the contributors and sends back the results into our directory structure.
- Special care has been taken to guarantee bit-by-bit accuracy on any computer platform. A checkpoint-restart mechanism is implemented as well.

SixTrack: Developments

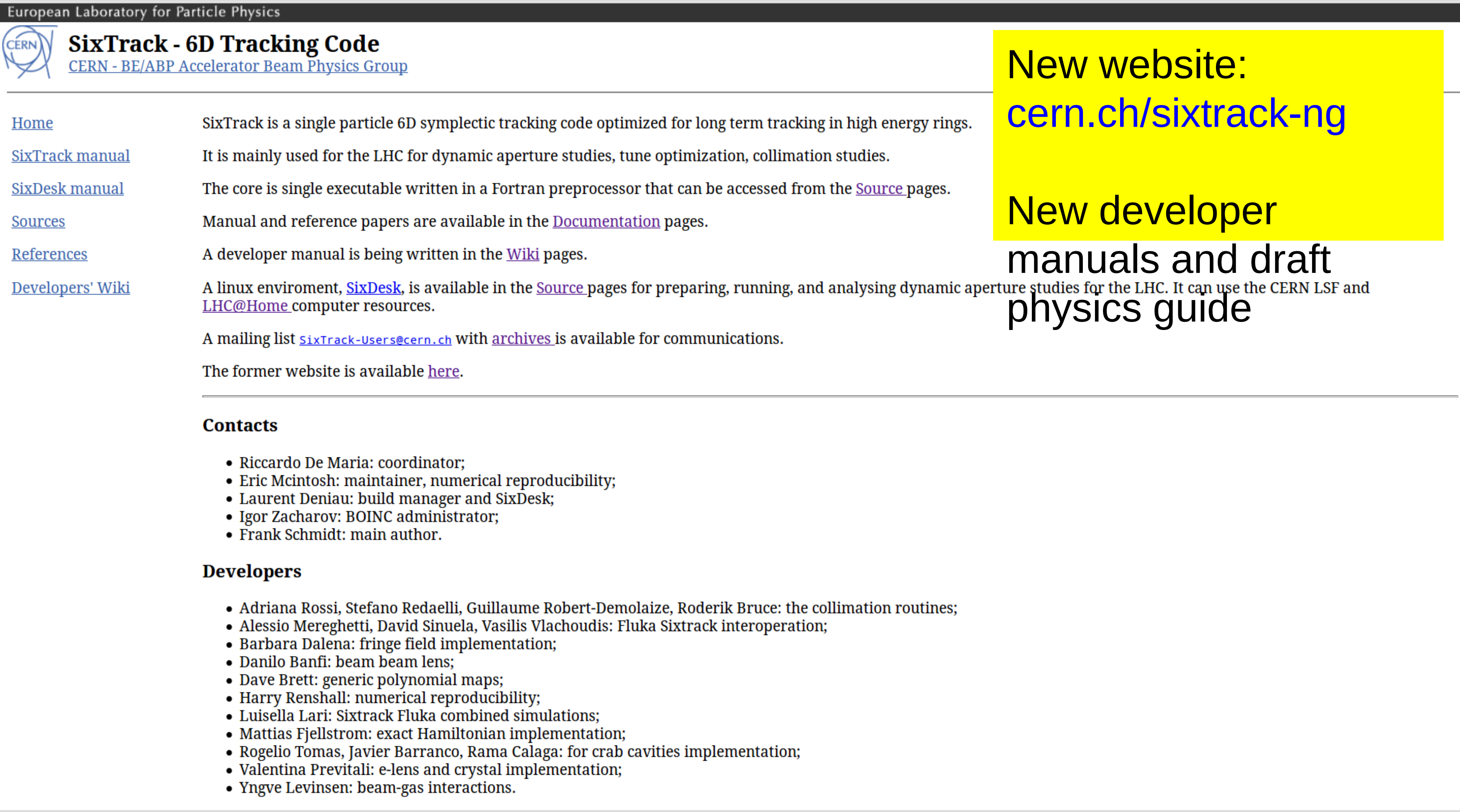
On going developments for short term goals:

- adding additional physical elements (exact drift and thin bends, arbitrary order rf-multipoles, fringe fields for quadrupoles and dipoles) on the existing code;
- review physics and developers' documentation;
- review of post processing routines: DA estimates, frequency/amplitude maps;
- introduction of new scattering routines in the collimation extension;
- improvements of the LHC run environment (SixDesk) to overcome AFS filesystem limitations using SQLite and MySQL databases;
- review of numerical accuracy for very long term simulations (numerical precision and reproducibility has been already addressed and solved by E. McIntosh);
- review of build system for cross -platform, -os, -compiler compilation;
- fluka inter-operation
- open source licensing of the code.

Long term goals (very desirable but no resources at the moment):

- decoupling of transfer maps, post processing, real coordinates and linear maps tracking, pre-processing (invariant search) to reduce complexity of the code and allow faster developments;
- Re-write the CPU intensive parts for parallel processors (CUDA, OpenCL, OpenMP technologies under evaluation) to gain in computational efficiency.

SixTrack: references



European Laboratory for Particle Physics

SixTrack - 6D Tracking Code
CERN - BE/ABP Accelerator Beam Physics Group

- Home
- SixTrack manual
- SixDesk manual
- Sources
- References
- Developers' Wiki

SixTrack is a single particle 6D symplectic tracking code optimized for long term tracking in high energy rings.

It is mainly used for the LHC for dynamic aperture studies, tune optimization, collimation studies.

The core is single executable written in a Fortran preprocessor that can be accessed from the Source pages.

Manual and reference papers are available in the Documentation pages.

A developer manual is being written in the Wiki pages.

A linux enviroment, SixDesk, is available in the Source pages for preparing, running, and analysing dynamic aperture studies for the LHC. It can use the CERN LSF and LHC@Home computer resources.

A mailing list SixTrack-Users@cern.ch with archives is available for communications.

The former website is available here.

Contacts

- Riccardo De Maria: coordinator;
- Eric Mcintosh: maintainer, numerical reproducibility;
- Laurent Deniau: build manager and SixDesk;
- Igor Zacharov: BOINC administrator;
- Frank Schmidt: main author.

Developers

- Adriana Rossi, Stefano Redaelli, Guillaume Robert-Demolaize, Roderik Bruce: the collimation routines;
- Alessio Mereghetti, David Sinuela, Vasilis Vlachoudis: Fluka Sixtrack interoperation;
- Barbara Dalena: fringe field implementation;
- Danilo Banfi: beam beam lens;
- Dave Brett: generic polynomial maps;
- Harry Renshall: numerical reproducibility;
- Luisella Lari: Sixtrack Fluka combined simulations;
- Mattias Fjellstrom: exact Hamiltonian implementation;
- Rogelio Tomas, Javier Barranco, Rama Calaga: for crab cavities implementation;
- Valentina Previtali: e-lens and crystal implementation;
- Yngve Levinsen: beam-gas interactions.

New website:
cern.ch/sixtrack-ng

New developer manuals and draft physics guide

Reserve

Some History I

- In 1989 I visited LBL to learn about Differential Algebra and NormalForm from Martin Berz and Etienne Forest respectively.
- As a consequence I fully "DA-ified" the SixTrack code including NormalForm and parameter dependence (k-values of any nonlinear family of elements).
- Anybody working on SixTrack (e.g. Rogelio!) knows that for any new element one has to modify the code at two code locations.
- In the DA part relevant lines start with the 4 letters "*FOX".
- In absence of overloading and polymorphic types in FORTRAN77 the code has to go through a pre-compiler to replace number operations by there equivalent DA operations.
- Despite this cumbersome approach SixTrack continues to work fine until these days.
- For detailed map analysis Etienne's original codes "dacom" and "dalie" are still available and continue to work for DA maps as produced by PTC. This works due to the fact that I have kept one of the DA map formats of PTC identical to the original one from 1989.
- In 2000 Etienne came for a long trip to CERN to upgrade SixTrack to his newest LIELIB package. M Berz's DA package was further optimized for speed.

Some History II

- In the early stages (1994) of the CLASSIC project led by John Irwin, both Etienne Forest and Johan Bengtsson participated but left pretty soon and predicting its failure. In retrospect it seems strange that this project has been started at all given the fact that the SSC project had been canceled a year earlier.
- Chris Iselin working on MAD9 was part of the CLASSIC team. It seems that he did most the work. In fact, it is not clear how much code has actually been delivered by other CLASSIC contributors.
- Early 2000 it became obvious that MAD9 was not ready to be used for the LHC optics design.
- During 2000 MAD-X was started as a crash program and I accepted to be the custodian October 2000.
- In the summer of 2000 Etienne and myself worked on SixTrack90, which was also called small code at some time. These codes have been prototypes for PTC (Polymorphic Tracking Code), baptized in March 2001.
- By that time the DA package and Etienne's Lielib code have been overloaded in Fortran90 and called FPP (Full Polymorphic Package).
- In October 2002 we had the first prototype PTC/FPP linked to MAD-X.
- On July 17th 2003 PTC became officially part of MAD-X including a tiny (73 lines) ptc_twiss module.

Project Style I

1. **Manpower Limitations**

   The good old times: MAD8: Hans Grote, MAD9: Chris Iselin, SixTrack: Frank Schmidt

   Today: MAD-X & SixTrack: Frank Schmidt (50%)

2. **Experience: Closer user involvement leads to better modules**

3. **Less stringent programming requirements (last transparency) allows the linking of various different modules**

➔ Organized Team work rather than a single expert