NEW FEATURES OF THE 2017 SixTrack RELEASE*

K. Sjobak†, R. De Maria, E. McIntosh, A. Mereghetti, CERN, Geneva, Switzerland
J. Barranco‡, EPFL, Lausanne, Switzerland; M. Fitterer, Fermilab, IL, USA
V. Gupta§, Indian institute of technology Guwahati, Gwahati, India
J. Molson, LAL, Univ. Paris-Sud, CNRS/IN2P3, Université Paris-Saclay, Orsay, France

Abstract

The SixTrack particle tracking code is routinely used to simulate particle trajectories in high energy circular machines like the LHC and FCC, and is deployed for massive simulation campaigns on CERN clusters and on the BOINC platform within the LHC@Home volunteering computing project. The 2017 release brings many upgrades that improve flexibility, performance, and accuracy. This paper describes the new modules for wire- and electron lenses (WIRE and ELEN), the expert interface for beam-beam element (BEAM/EXPERT), the extension of the number of simultaneously tracked particles, the new Frequency Map Analysis (FMA) postprocessing option, the generation of a single zip of selected output files (ZIPF) in order to extend the coverage of the studies in LHC@HOME (e.g. FMA and on-line aperture checks), coupling to external codes (DYNK-PIPE and BDEX), a new CMAKE based build- and test mechanism, and internal restructuring.

INTRODUCTION

SixTrack is a 6D symplectic single-particle tracking code written in Fortran, C, and C++ [1–3]. It is routinely used for studying the dynamic aperture and collimation system of high-energy circular machines like the LHC, FCC, and SPS. The code runs on a large number of operating systems (Linux, OS X, Free BSD, Net BSD, and Windows), compilers (gfortran, ifort, and nagfor), and machine architectures (x86, x86_64, and aarch64), with numerical results being exactly reproduced across all of them [4]. This enables us to run studies on local machines, local clusters, or using LHC@HOME with BOINC [5, 6].

Since the last BOINC release (version 4.5.17 from May 2014), several new features have been developed for SixTrack, and many of those are summarized in this paper. Other changes are described in separate papers, such as the dynamic kicks (DYNK) module [7, 8], and on-line aperture checking [9]. The usage of these new features are described in the SixTrack user manual [10], which is maintained in sync with and in the same git repository as the code.

BEAM LINE ELEMENTS

Electron Lens

A new flexible input block for electron lenses used e.g. for halo control or beam-beam compensation [11] has been implemented, allowing the definition of electron lenses with different electron beam distribution [12]. Currently only the hollow electron lens is defined [13], however the structure of the code would make it easy to add other types. Furthermore, the parameters of the electron lens can be modified on a turn-by-turn basis through DYNK, allowing simulation of operation with the electron lens in a modulated mode.

Wire

Current bearing wires are considered as an elegant option for the compensation of long-range beam-beam interactions in the LHC [14–16]. To simulate the effect of the wire in combination with beam-beam interactions for the LHC, a new symplectic kick map has been developed for the wire and implemented in SixTrack [13].

Improved Beam-Beam Lens

An optional "EXPERT" mode has been defined for the BEAM block. This gives more freedom when defining the parameters for the 6D beam-beam lens [17] than previously possible, getting the Σ matrix for the strong beam from the input file instead of assuming that the strong and weak beam have certain symmetries and using the internally calculated parameters for the weak beam. In particular, it is possible to simulate arbitrary longitudinal distributions and orbit distortions (e.g. crab crossing with RF curvature or asymmetric beam sizes at the IP) of the strong beam by defining a series of 6D slices with pre-computed parameters.

The input parameters can be computed and exported from MAD-X [18] using the MAD-X to SixTrack converter. Furthermore, since the parameters are now taken directly from the input file, implementing the possibility to change the parameters of the strong beam on a turn-by-turn basis using DYNK is now within reach.

At the same time, the "traditional" interface is still available. To ease conversion, it now writes out the input for the "EXPERT" interface that would exactly reproduce the same results.

OUTPUT

In order to be able to bring back more data from e.g. BOINC, a new option ZIPF has been added. This allows the user to specify a list of output files that should be packed in an archive called Sixout.zip, which can then be returned

* Research supported by the HL-LHC project
† kyrre.ness.sjoebaek@cern.ch
‡ This work is supported by the European Circular Energy-Frontier Collider Study, H2020 Framework Programme under grant agreement no. 654305
§ Work supported by Google Summer of Code 2016

from the volunteer or the batch nodes. New features such as Frequency Map Analysis and the online aperture will use this feature to bring back the results.

Furthermore, a module "DUMP" for exporting the particle distribution every *n* turns at any or all point(s) in the lattice has been added. This makes the results easier to post-process and visualize in external programs. Several output formats are possible, and adding more require only trivial code changes.

FREQUENCY MAP ANALYSIS

The Frequency Map Analysis (FMA) is in general a powerful tool for the analysis of non-linear dynamics based on the calculation of the diffusion in tune versus the initial tune of the particle or its normalized amplitude. The FMA analysis currently implemented in SixTrack first reads the particle distributions exported from DUMP, then normalizes the particle amplitudes following the formalism described in [19]. From this, it calculates the tune of each particle using a version of the PLATO library [20] modified for double precision and exact reproducibility, which provides different methods for the tune calculation. The diffusion in tune or amplitude can then be visualized using the SixDeskDB [21] extension of SixTrack's post-processing tool SixDesk [22].

INCREASED PARTICLE LIMITS

Formerly SixTrack was limited to tracking only 64 particles at a time, this limit has now been increased to 65'536 particles. The old limit is sufficient for dynamic aperture studies with a large number of turns, as the computing time for each turn is kept relatively short; covering a larger number of phase-space points is done by launching multiple simulations. However this limit is inefficient when running simulations where many particles are tracked over fewer turns, such as for collimation simulations. Special measures are therefore taken in the collimation code to get around this problem, calling the tracking loop multiple times and resetting the particle distributions and DYNK each time. However this necessitates extra initialization, slowing down the computation. Another case where this limit is problematic is for studying collective effects, which can be done by coupling to external codes.

The main limitation preventing increasing the particle number was that the data which is used in post-processing was written to a separate file for each each particle pair. Additionally, the arrays holding second order maps for thick tracking have been moved to allocated arrays, avoiding excessive memory usage when doing thin tracking with large particle numbers. It is now optionally possible to write this data to a single file, which unlocks the possibility to track more particles.

GENERALIZED COUPLING TO EXTERNAL CODES

A general module for coupling to external codes is under preparation, allowing the external code to change element properties in the simulation on a turn-by-turn basis (DYNK-PIPE), and to read and exchange particles at any point(s) in the ring (BDEX). This allows evaluating the effects of e.g. crab cavity beam loading and self-consistent simulations of crab cavity failure scenarios [23], and to couple to external scattering programs such as FLUKA. The BDEX block has a significant feature overlap with the existing FLUKA coupling [24], however it is intended to be more general, being able to connect to multiple codes and easy to extend to new cases.

INTERNAL CHANGES

In addition to the new features available to SixTrack users, much has been changed behind the scenes. These changes make the code easier to maintain, and reduce the effort necessary to modify or understand the internals of the code. Some of the most important of these changes are mentioned below. Furthermore, the documentation has been improved, and explicit instructions for building the code across all supported platforms and configurations have been provided [25].

Code Refactoring

In order to make DYNK possible, a unified framework for initializing an element after changing its settings was needed. This was produced as the subroutine `initialize_element`, which is called at the start of the simulation to propagate settings from the internal representation of the input files to the variables used in the tracking, and then called again after changing the in-memory input files through DYNK.

For the checkpoint/restart (C/R) version used for e.g. BOINC, the "standard output" of the code is first written to a temporary file and then later flushed to the actual output file `fort.6`, while for the non-C/R version the output is written directly to standard output. Before Fortran 2003, the language provided no standardized way of looking up the standard output unit, making it necessary to duplicate all write statements. By using the `ISO_FORTRAN_ENV` module from Fortran 2003, these almost-duplicate code lines could be removed, saving a total of 8000 lines of code, and greatly improving readability of many parts of the code.

The SixTrack code uses two preprocessors, one of which is `astuce`. This program plays much of the same role as the C preprocessor, gluing together the code seen by the compiler from several call blocks similar to `#include` in C. Unfortunately, the internals of this Fortran code were poorly understood and not documented, and the program itself was fragile to errors in its input, often silently producing corrupted results. A rewrite (in C++) was therefore done; the new version `astuce++` has the benefit of not just being more robust, but also making it possible to have long identifiers for the call block names. It also makes it possible to combine call blocks (`+cd`) from several source files (`.s`), which will both reduce code duplication and make it possible to split the currently mostly monolithic (single file) code into several more manageable source files.

Parts of the interface code for BOINC was re-factored; the most important change here was to drop the input/output zip/unzip functionality built into BOINC and instead use the open-source library LibArchive [26]. The same library was also used to implement ZIPF, described above.

The collimation section of the code, which should soon be available on BOINC, has been split from a large and mostly monolithic block into a number of smaller functions. These are called at set points in tracking, for example before each collimator, after each turn, etc. This has greatly simplified the insertion of the interaction physics models from the codes Merlin [27, 28] and Geant4 [29], allowing the user to select their desired interaction model for collimation at compile time.

Build and Test System

The build system needs to be able to handle many and complex combinations of compile time feature options, compilers, and operating systems. Furthermore, the build process uses two non-standard pre-processors (`astuce` and `dafor`) which are chained to produce and transform the actual compiled source files. Furthermore, multiple programming languages and language versions are used throughout the project. A shell- and makefile based system was previously used to build the code, however it was hard to maintain and to extend, and it used completely separate and partially duplicate code paths for several combinations of options, compilers, and OSs. A purely make-based build system was also written, however it ended up becoming overly complicated in order to meet the requirements. A CMAKE based build system was therefore written, which is conceptually similar to the original shell- and make based system, generating a specialized makefile for each option combination. It is however much more maintainable, easy to extend, works on all target platforms, and supports parallel compilation which significantly speeds up the build process.

A major benefit of a CMAKE based build system is the built-in integration with CTEST. This makes running the test suite automatic, making it easier to use and more useful. This test suite covers a significant part of the code with 59 tests, and can also exercises the checkpoint/restarting and BOINC input loading capability in each test. Furthermore, tests results may be submitted to CDASH, which collects and visualizes test results between different machines, platforms etc. on a centralized webpage [30]. Finally, CTEST makes it possible to collect code coverage information, which shows which code lines have been exercised by the test suite; this information is also displayed on the CDASH page.

Source Control with Git

The hosting of the SixTrack sources was moved from CERN's SVN service to GitHub [31]. This simplifies the development process, especially with multiple contributors working on different sub-projects at different pace, and many developments ongoing in parallel. For this, the project's SVN history (which stretched back to version 4.0 from 2005) was converted to the new format, keeping all the metadata available after the transition. In addition to the excellent support for branching and merging, git's support for submodules is very useful for handling the external libraries linked from SixTrack.

For hosting the main repository, we also considered using CERN's private GitLab service, however in the end we decided to use GitHub in order to make it easier for external collaborators, because GitHub is more widely used, and because the fork-and-pull model is a better fit for the project's typical workflow. Since SixTrack is an open source project, there were no drawbacks to using such an external service where the code must be publicly shown; furthermore since each developer has a git clone which contains the entire repository information, there is no risks of loosing the source code or version history should GitHub suddenly become unavailable. Finally, GitHub's features encourages and helps structure the discussion about code changes and issues.

CONCLUSIONS

As described in this paper, a large number of changes have been made to SixTrack since the last BOINC release. These new features will enable many new studies in the near future, both for the current LHC, HL-LHC, FCC, SPS, and several other machines. Furthermore, there have been several internal changes that makes the code easier to maintain, allowing the development at an increased speed and decreased error rate.

ACKNOWLEDGEMENTS

The BOINC volunteers of LHC@home are warmly thanked for their contribution of CPU-time and especially for helping testing the new executables over a large range of platforms which were not easily available to the authors.

REFERENCES

[1] F. Schmidt, "SixTrack Version 4.2.16 Single Particle Tracking Code Treating Transverse Motion with Synchrotron Oscillations in a Symplectic Manner", CERN/SL/94-56, 2012.

[2] G. Ripken and F. Schmidt, "A symplectic six-dimensional thin-lens formalism for tracking", DESY 95–63 and CERN/SL/95–12(AP), 1995.

[3] R. De Maria *et al.*, "Recent developments and future plans for SixTrack", in *Proc. 4th Int. Particle Accelerator Conf. (IPAC'13)*, Shanghai, China, May 2013, paper MOPWO028, pp. 948–950.

[4] E. McIntosh, paper in preparation.

[5] M. Giovannozzi *et al.*, "LHC@Home: A volunteer computing system for massive numerical simulations of beam dynamics and high energy physics events" in *Proc. 3rd Int. Particle Accelerator Conf. (IPAC'12)*, New Orelans, LA, USA, May 2012, MOPPD061, pp. 505–507.

[6] A. Mereghetti *et al.*, "Recent Development Activities of SixDesk, the Simulation Environment for SixTrack", presented at the 8th Int. Particle Accelerator Accelerator Conf. (IPAC'17), Copenhagen, Denmark, May 2017, paper THPAB045, this conference.

[7] K. Sjobak, H. Burkhardt, R. De Maria, A. Mereghetti, and A. Santamaría García, “General functionality for turn-dependent element properties in SixTrack”, in *Proc. 5th Int. Particle Accelerator Conf. (IPAC’15)*, Richmond, VA, USA, May 2015, paper MOPJE069, pp. 468–471.

[8] K. Sjobak, “Dynamic simulations in SixTrack”, in Proceedings from the HL-LHC collimation workshop 2015, CERN, to be published.

[9] A. Mereghetti *et al.*, “SixTrack for Cleaning Studies: 2017 Updates”, presented at the 8th Int. Particle Accelerator Accelerator Conf. (IPAC’17), Copenhagen, Denmark, May 2017, paper THPAB046, this conference.

[10] F. Schmidt, R. De Maria, M. Fitterer, K. Sjobak, *et al.*, “SixTrack version 4.6.16 – Single Particle Tracking Code Treating Transverse Motion with Synchrotron Oscillations in a Symplectic Manner – User’s reference manual” http://sixtrack.web.cern.ch/SixTrack/doc/manual_dev/six.pdf

[11] V. Shiltsev, “Electron Lenses for Super-Colliders”, Springer, 2016

[12] M. Fitterer *et al.*, “Implementation of Hollow Electron Lenses in SixTrack and first simulation results for the HL-LHC”, presented at the 8th Int. Particle Accelerator Accelerator Conf. (IPAC’17), Copenhagen, Denmark, May 2017, paper THPAB041, this conference.

[13] R. De Maria *et al.*, “SixTrack Physics Manual”, http://sixtrack.web.cern.ch/SixTrack/doc/physics_manual/sixphys.pdf

[14] J.-P. Koutchouk, “Principle of a Correction of the Long-Range Beam-Beam Effect in LHC using Electromagnetic Lenses”, LHC Project Note 223, 2000.

[15] J.-P. Koutchouk, “Correction of the Long-Range Beam-Beam Effect in LHC using Electromagnetic Lenses”, SL Report 2001-048, 2001.

[16] S. Fartoukh, A. Valishev, Y. Papaphilippou, and D. Shatilov, “Compensation of the long-range beam-beam interactions as a path towards new configurations for the high luminosity LHC”, Phy. Rev. ST Accel. Beams, vol. 18, p. 121001, Dec. 2015.

[17] L.H.A. Leunissen *et al.*, “Six-dimensional beam-beam kick including coupled motion”, Phys. Rev. ST Accel. Beams, vol. 3, December 2000, p. 124002.

[18] MAD-X Project website, http://cern.ch/mad

[19] F. Willeke and G. Ripken, “Methods of Beam Optics", DESY 88-114, 1988.

[20] M. Giovannozzi, E. Todesco, A. Bazzani, and R. Bartolini, “PLATO: a program library for the analysis of nonlinear betatronic motion”, Nucl. Instrum. and Methods A, vol. 388, 1996.

[21] SixDeskDB GitHub page, https://github.com/SixTrack/SixDeskDB

[22] E. McIntosh, and R. De Maria, “The SixDesk Run Environment for SixTrack”, CERN-ATS-Note-2012-089 TECH, 2012.

[23] R. Apsimon, G. Burt, A.C. Dexter, P. Baudrenghien, K. Sjobak, and R. Appleby “Modelling the Low Level RF Response on the Beam During Crab Cavity Quench”, presented at the 8th Int. Particle Accelerator Accelerator Conf. (IPAC’17), Copenhagen, Denmark, May 2017, paper MOPVA102, this conference.

[24] A. Mereghetti *et al.*, “SixTrack-FLUKA active coupling for the upgrade of the SPS scrapers”, in *Proc. 4th Int. Particle Accelerator Conf. (IPAC’13)*, Shanghai, China, May 2013, paper WEPEA064, pp. 2657–2659.

[25] K. Sjobak and J. Molson, “Compiling, building, and testing SixTrack”, http://sixtrack.web.cern.ch/SixTrack/doc/building_sixtrack/building_sixtrack.pdf

[26] libarchive “Multi-format archive and compression library” website, https://www.libarchive.org/.

[27] R.B. Appleby, R.J. Barlow, J.G. Molson, M. Serluca, and A. Toader, “The practical Pomeron for high energy proton collimation” Eur. Phys. J. C, vol. 76, p. 520, October 2016.

[28] J. Molson *et al.*, “A Comparison of Interaction Physics for Proton Collimation Systems in Current Simulation Tools”, presented at the 8th Int. Particle Accelerator Accelerator Conf. (IPAC’17), Copenhagen, Denmark, May 2017, paper WEOBA1, this conference.

[29] S. Agostinelli *et al.*, “Geant4 – a simulation toolkit”, Nucl. Instrum. and Methods A, vol. 506, p. 250–303, July 2003.

[30] CDash dashboard for SixTrack, http://abp-cdash.web.cern.ch/abp-cdash/index.php?project=SixTrack

[31] SixTrack GitHub page, https://github.com/SixTrack/SixTrack