ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAF02

SixTrack PROJECT: STATUS, RUNTIME ENVIRONMENT, AND NEW DEVELOPMENTS*

R. De Maria†, J. Andersson, V. K. Berglyd Olsen, L. Field, M. Giovannozzi, P. D. Hermes, N. Høimyr, S. Kostoglou, G. Iadarola, E. Mcintosh, A. Mereghetti, J. Molson, D. Pellegrini, T. Persson, M. Schwinzerl, CERN, Geneva, Switzerland
E. H. Maclean, CERN, Geneva, Switzerland and University of Malta, Msida, Malta
K. N. Sjobak, CERN, Geneva, Switzerland and University of Oslo, Oslo, Norway
I. Zacharov, EPFL, Lausanne, Switzerland
S. Singh, IIT Madras, India‡

Abstract

SixTrack is a single-particle tracking code for high-energy circular accelerators routinely used at CERN for the Large Hadron Collider (LHC), its luminosity upgrade (HL-LHC), the Future Circular Collider (FCC), and the Super Proton Synchrotron (SPS) simulations. The code is based on a 6D symplectic tracking engine, which is optimised for long-term tracking simulations and delivers fully reproducible results on several platforms. It also includes multiple scattering engines for beam-matter interaction studies, as well as facilities to run integrated simulations with FLUKA and GEANT4. These features differentiate SixTrack from general-purpose, optics-design software like MAD-X. The code recently underwent a major restructuring to merge advanced features into a single branch, such as multiple ion species, interface with external codes, and high-performance input/output (XRootD, HDF5). This restructuring also removed a large number of build flags, instead enabling/disabling the functionality at run-time. In the process, the code was moved from Fortran 77 to Fortran 2018 standard, also allowing and achieving a better modularization. Physics models (beam-beam effects, RF-multipoles, current carrying wires, solenoid, and electron lenses) and methods (symplecticity check) have also been reviewed and refined to offer more accurate results. The SixDesk runtime environment allows the user to manage the large batches of simulations required for accurate predictions of the dynamic aperture. SixDesk supports CERN LSF and HTCondor batch systems, as well as the BOINC infrastructure in the framework of the LHC@Home volunteering computing project. SixTrackLib is a new library aimed at providing a portable and flexible tracking engine for single- and multi-particle problems using the models and formalism of SixTrack. The tracking routines are implemented in a parametrized C code that is specialised to run vectorized in CPUs and GPUs, by using SIMD intrinsics, OpenCL 1.2, and CUDA technologies. This contribution presents the status of the code and an outlook on future developments of SixTrack, SixDesk, and SixTrackLib.

* Research supported by the HL-LHC project
† riccardo.de.maria@cern.ch
‡ Work supported by Google Summer of Code 2018

INTRODUCTION

SixTrack [1,2] is a 6D single-particle symplectic tracking code able to compute the trajectories of individual relativistic charged particles in circular accelerators for studying dynamic aperture (DA) or evaluating the performance of beam-intercepting devices like collimators [3]. It can compute linear and non-linear optics functions, time-dependent effects, and extract indicators of chaos from tracking data. SixTrack implements scattering routines and aperture calculations to compute "loss maps", i.e., leakage from collimators as a function of longitudinal position along the ring, and collimation efficiency [4].

Different from a general-purpose code like MAD-X [5,6], SixTrack is optimised for speed and numerical reproducibility. It can be also linked with the BOINC library to use the volunteering computing project LHC@Home [7]. SixTrack studies, such as estimation of dynamic aperture of large storage rings like the Large Hadron Collider (LHC) or the Future Circular Collider (FCC), require massive computing resources, since they consist of scans over large parameter spaces for probing non-linear beam dynamics over long periods.

The SixDesk runtime environment manages SixTrack simulations from input generation, job queue management (using HTCondor or LSF in the CERN BATCH service and customised software in CERN Boinc server), to collecting and post-processing results.

SixTrackLib is a new library built from scratch in C with the main aim of offering a portable tracking engine for other codes and offloading SixTrack simulation to GPUs.

This paper summarises the main existing features of SixTrack, SixDesk and SixTrackLib and provide detail about the main development lines.

MAIN FEATURES

SixTrack tracks an ensemble of particles defined by a set of coordinates through several beam-line elements, using symplectic maps [8–10], or scattering elements.

Coordinates

The set of coordinates is larger than the minimum needed to describe the motion. Additional variables are used to store energy-related quantities used in the tracking maps

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAF02

that are updated only on energy changes, which does not occur very frequently in synchrotrons in absence of radiation effects, to save computational time. Thick maps for dipole and quadrupoles also reuse the energy-dependent factors of the first- and second-order polynomial of the map that are recalculated at each energy change. Furthermore, different ion species, such as debris from interaction with matter, can be tracked at the same time using an extension of the usual symplectic formalism [11].

Variables used internally in tracking are not canonical, however, once they are converted to canonical form, the maps are symplectic. Different from other codes, SixTrack uses

$$\sigma = s - \beta_0 ct$$

as the longitudinal coordinate during tracking to avoid rounding errors associated to the relativistic β when updating time delays in drifts and

$$\left(\zeta = \frac{\beta}{\beta_0}\sigma, \quad \delta = \frac{P - P_0}{P_0}\right)$$

as conjugate canonical variables in 6D optics calculations which use explicitly symplectic maps.

Beam-line Elements

Table 1 shows the different types of beam-line elements implemented in SixTrack. Thin multipoles are used in conjunction with the `MAKETHIN` and `SIXTRACK` commands in MAD-X to implement symplectic integrators of thick maps. Thin multipoles include the effect of the curvature, when present, up to the second order. The tracking maps have been recently reviewed and benchmarked against MAD-X and its optics module for consistency.

Table 1: Physical Elements Implemented in SixTrack

Drift expanded	Drift exact [12]
Single thin multipole	Thin multiple block
Thick dipole-quadrupole	Thin solenoid
Accelerating cavities	RF-multipoles [13]
4D Beam-beam	6D beam-beam [14]
Wire [15]	Hollow electron lens [16, 17]

Scattering

SixTrack embeds the K2 scattering engine [18, 19], capable of simulating the basic scattering processes undergone by an ultra-relativistic proton in the multi-TeV range when passing through matter. The simulated processes range from ionisation energy loss and multiple Coulomb scattering to point-wise interactions like Coulomb, elastic, and inelastic events, including single diffractive scattering. Compound materials of interest for the low-impedance upgrade of the LHC collimators are implemented via averaged nuclear and atomic properties [20]. Other scattering models can be imported and made available in the SixTrack executable, such as that of Merlin [21] and Geant4 [22, 23].

A new scatter block is under development to offer a general framework for simulating scattering events in SixTrack. Currently, it supports beam scattering against a target specified as an area density distribution at a thin marker inserted into the lattice. Internally, the scattering module supports elastic scattering through Monte Carlo sampling of experimental data from Totem. Alternatively, scattering events can be generated on the fly by Pythia8 [24], in which case elastic and diffractive processes are supported.

Optics Calculations

SixTrack contains matrix code for 5D optics calculation and a 6D tracking engine using Truncated Power Series Algebra library (TPSA [25]) for 6D optics calculation. The 6D tracking engine uses canonical variables and it provides a cross-check of the symplecticity of the one-turn-map. Coupled Twiss parameters (using the Mais-Ripken formalism [26]) can be extracted along the lattice. The optics parameters are optionally used in the beam-beam elements for self-consistent simulations. The 6D optics module has been recently improved by removing some unnecessary ultra-relativistic approximations which introduced small symplectic errors.

Dynamic Effects

A general functionality for dynamically-changing simulation settings on a turn-by-turn basis has been implemented [27,28]. This allows setting magnet strengths including multipoles, RF amplitude and phase, reference energy, and beam-beam element as a function of turn number. This can be useful for a number of different studies, e.g. magnet snap-back in the LHC [29], HL-LHC crab cavity failure scenarios [30–32], studies of beam losses during energy ramp [33], and hollow electron-lens modulation [34]. The settings can be internally computed as a function of turn number, or loaded from a file. These functions are specified using a flexible language that allows combining functions to achieve the required effect. The architecture of the functionality makes it easy to add support for new elements or new functions.

Post Processing

Long-term tracking with SixTrack is used extensively at CERN for studying the DA, with a typical study consisting of up to $\sim 2 \times 10^6$ individual tracking simulations over $10^5 - 10^6$ turns (see Figure 1 for an example).

Tracking data are post processed during the study and summary files, containing the main results of the simulation for each initial condition, are returned back to the user. In particular, tracking summary files for each initial condition identify particle loss/survival, final surviving turn numbers and the inferred particle amplitudes.

Particle's invariants are calculated for each initial condition and based on the average invariant over a user-defined range of turns. An initial estimate of the invariant is obtained by assuming no coupling between the planes of motion, via the usual relation for the Courant-Snyder ellipse, e.g. for the

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAF02

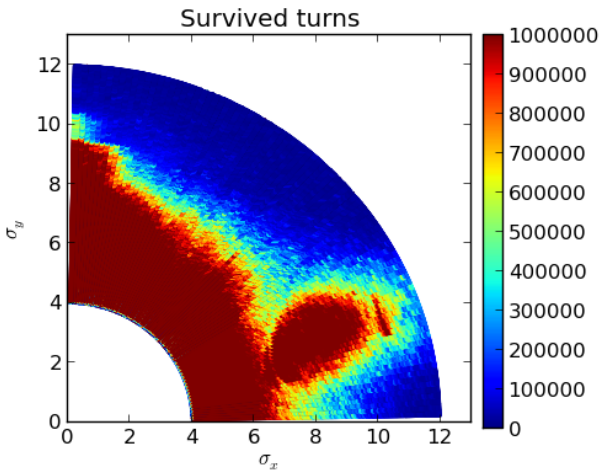

Figure 1: Survival plot of a fine phase-space scan for the LHC. The simulations was divided in task of 60 particles pairs covering the phase space in 2σ and 1.5° steps.

horizontal plane. Alternatively, an estimate of the decoupled single-particle emittance for the three oscillation modes can be calculated from the eigenvectors of the motion ($\bar{v}$), which may be constructed from the one-turn map, see for example [26]. Various parameters relevant to the nonlinear motion, such as smear and detuning, are also evaluated.

In addition to quantities relevant to particle survival, estimates of the long-term stability are obtained through a Lyapunov-like analysis performed by examining the phase-space separation of initially close by particle pairs. In particular, the angular separation in phase for the three oscillation modes

$$\frac{1}{\pi}\sqrt{\frac{\Delta\phi_1^2+\Delta\phi_2^2+\Delta\phi_3^2}{N}},$$

where $N = (1,2,3)$ for $(2D,4D,6D)$ motion, respectively, is considered. Linear fits to the logarithm of the separation as function of the logarithm of the turn number identify the maximum separation rate between each particle pair. This quantity is returned, along with the maximum separation in phase, to provide approximate indicators of the onset of chaotic motion in place of the far more computationally intensive Lyapunov exponents [35, 36].

Summary files for the outcome of each initial condition are collectively post-processed by the user using external tools, in order to identify minimum boundaries in the (σ_x, σ_y) space for particle survival over the tracked number of turns, as well as to study the evolution of DA as a function of the turn number.

Frequency Analysis

A collection of routines for frequency analysis has been linked in SixTrack, namely PLATO [37] and a C++ implementation of the Numerical Analysis of Fundamental Frequency (NAFF) algorithm has been developed [38]. These algorithms allow for a more refined, compared to plain FFT, tune determination with a much faster convergence, i.e. requiring a shorter number of turns. By comparing the tune determination at different time intervals, diffusive frequency maps can be been computed [39]. With the resolution of the frequency map, resonance lines become visible, even in the case of a tune modulation from a quadrupolar ripple (the triplets in IR1 and 5) with frequency of 550 Hz and relative amplitude of 10^{-7}, as shown in Figure 2.

Input and Output

Initial conditions can be given in amplitude steps or taken from an external file. A dump module offers multiple ways to extract tracking data both in terms of type observable (physical coordinates, canonical coordinates, normalised coordinates, averages and first order distribution momenta) in a selection of turns and observation points. Data are written in ASCII and, in a few cases, a binary option is also available. Support for output of simulation data to a HDF5 [40] files and ROOT [41] is also currently being developed.

Furthermore, it is planned to develop a new way to generate the distribution that is used as initial conditions for tracking. This will provide the functionality to create matched or mismatched distributions in both physical and normalised coordinates.

Interfaces to External Programs

Collimation studies can also be performed running SixTrack coupled [42] to Fluka [43, 44]. In this configuration, the two codes exchange particles at run time, with the aim of combining the refined tracking through the accelerator lattice, performed by SixTrack, with the detailed scattering models, implemented in Fluka, when the beam reaches intercepting devices. The use in the Fluka-SixTrack coupling of the same Fluka geometries used for subsequent energy-deposition calculations run with Fluka allows an excellent level of consistency of results.

Additionally, a more generalised interface "BDEX" for interfacing external codes is also included, enabling for example tracking of multiple bunches or coupling to cavity simulation codes. Here, the exact protocol can be implemented as a plug-in to SixTrack [45].

Building and Testing

A CMake-based build and test system has recently been added [45]. This greatly simplified the maintenance of the dependencies between the various build options, as well as the setup for building on the large range of supported platforms.

The testing framework CTest is also provided as part of CMake. For SixTrack, this is used to verify that the executables are still providing the expected output after the code has been modified, and to track the changes to the output. Furthermore, it is used for checking that the results from versions compiled for different platforms are in agreement, which is vital for BOINC. The main benefit of using CTest is that test running is fully automatic and gives a simple

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAF02

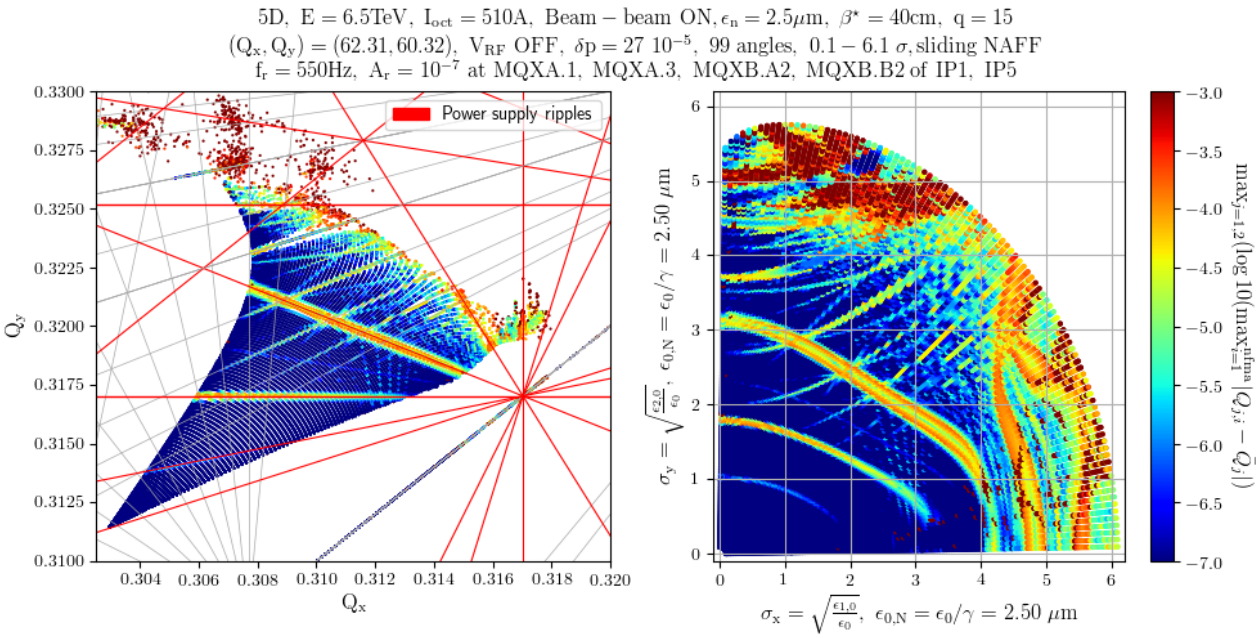

Figure 2: Frequency map using the NAFF method for LHC in the presence of a tune modulation.

pass/fail output, which is also integrated with GitHub for pull requests.

Performance

Thanks to the recent re-factoring, the internal particles arrays are fully dynamic, therefore the number of particles that can be tracked in parallel is limited by the system memory and not by a build-time flag. A machine model like the LHC, using about 18k elements and 4.6k high-order multipole blocks, needs about 220 µs per particle per turn on a single CPU core at 3.4 GHz. Typical studies requires of the order of 10^9–10^{12} particle turns and even more for parameter scans. For this reason, SixTrack is often used in conjunction with high-performance computing facilities described in the following section.

RUNTIME ENVIRONMENTS

The SixDesk environment [46] is the simulation framework used to manage and control the large amount of information necessary for, and produced by, SixTrack studies of dynamic aperture. It supports CERN batch systems [47] as well as the BOINC platform for volunteering computing available at the LHC@Home project [7]. The SixDeskDB post-processing tool collects data from SixDesk, performs post-processing analysis, and prepares reports and plots. It also offers a Python API for interactive analysis. Similarly to the SixTrack code, the SixDesk environment and SixDeskDB are continuously updated, extending the coverage of the studies and keeping the environment up to date with the latest developments in the CERN IT infrastructure.

LHC@Home and the CERN Batch System

Volunteer computing has been used successfully at CERN since 2004 with the LHC@Home project; it has provided additional computing power for CPU-intensive applications with small data sets, as well as an outreach channel for CERN activities. LHC@Home started off with SixTrack, which had been successively ported from mainframe to supercomputer to emulator farms and PCs. In order to run on the largest number of volunteer computers, SixTrack is compiled for the most common operating systems, architectures, and CPU instruction sets.

In terms of computing power provided by the volunteers to LHC@home, the system is capable of handling 1×10^5 tasks on average, with peaks of 3.5×10^5 tasks simultaneously running on 2.4×10^4 hosts observed during SixTrack intense simulation campaigns (see Figure 3). Every SixTrack task is run twice to eliminate random host errors and minimise the impact of a failing host. The LHC@Home capacity available for SixTrack can be compared to the average of 2.5×10^5 running tasks on 1.4×10^5 processor cores in the CERN computer centre, which is fully loaded with tasks from analysis and reconstruction of collisions recorded by LHC experiments, and has limited spare capacity for beam dynamics simulations.

The CERN batch system is presently managed by means of the HTCondor [48] package. Contrary to BOINC, most suitable for a steady stream of work units, the CERN batch system provides users with a responsive computing resource. Also, contrary to LHC@Home, no redundancy is implemented during task submission since the code run in a controlled environment, although very rarely hardware errors do appear in the results.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAF02

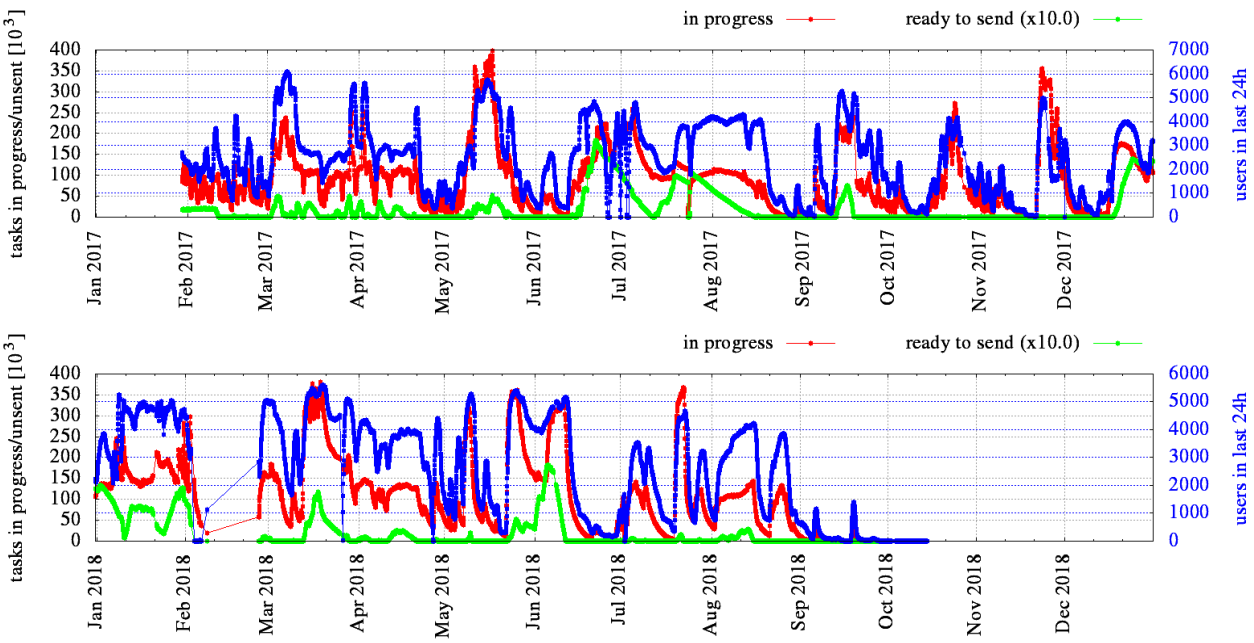

Figure 3: Summary of tasks and users during the last two years (upper frame: 2017; lower frame: 2018 to date) of the SixTrack application in the LHC@Home project. The number of users tends to increase, with the number of tasks absorbing most of the time all pending tasks.

Developments

One of the main development lines of SixDesk is porting the collimation studies to the BOINC platform for volunteer computing. This entailed a thorough revision of the collimation part, currently on-going, to make results numerically stable and reproducible across platforms. The possibility to interrupt and restart the computation (check-point/restart capability), which is essential to run on BOINC, is being added as well.

Other lines of development include: the addition of new parameters for dedicated scans of dynamic aperture; the possibility of running chains of jobs in BOINC, for simulating extended periods of beam time in the ring; a pre-filtering stage of submission to the CERN batch system prior to submission to BOINC, to avoid short tasks, with consequent inefficient use of volunteer resources, like bandwidth and time.

SixTrackLib

In the context of single-particle simulations, tracking requires no interaction between the calculations carried out for particle p_i and $p_{j\neq i}$ from a set of N particles. The memory requirements for representing each p_i typically ranges from 10^1 to 10^3 Bytes. The machine description can, over a single turn, be considered constant, although different elements and sections of the ring require a different amount of local resources. Still, SixTrack presents itself as an ideal candidate for a parallel implementation: strongly CPU bound with inherent parallelism and resource requirements not inherently scaling with the number of parallel processes.

Introducing parallelism into a mature code-base like SixTrack, even from such a favourable starting point, is challenging. It entails a high levels of complexity due to competing paradigms and concepts of parallel computing. In particular, a fast-changing technological landscape in combination with a diverse, multi-vendor and long-tailed selection of hardware available via initiatives like LHC@Home [7], as well as the realities of limited development resources are the main decision-making factors. These and other constraints motivated the design and ongoing development of a new, stand-alone library providing the core functionality of SixTrack.

SixTrackLib [49] is an open-source library developed from scratch using C and C++, allowing users to off-load the particle tracking onto supported HPC resources. As of this writing, it provides a) a representation for a set of particles; b) a set of beam-elements (drifts, multipoles, cavities, 4D and 6D beam-beam elements, etc.); c) a set of maps describing the tracking of the particles over the beam-elements; d) a dedicated generic buffer for managing and transferring data to the computing nodes; e) implementations and abstractions for different computing environments (auto-vectorized CPU code, OpenCL [50], CUDA [51]); f) high-level APIs in C, C++ and (under development) Python.

The chosen design allows for the complete separation of business logic from the modelled physics, allowing the latter to be shared across all architectures. Maps and tracking functions are implemented in a sub-set of the C99 language, in terms of the provided abstractions. The physics parts are exposed to the user in a modular and header-only fashion, allowing to tap into SixTrackLib under a wide-range of use cases currently out of reach for a stand-alone application such as SixTrack.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAF02

First simulations and tests prove the feasibility of the approach and reproduce results delivered by SixTrack. Performance analysis confirms that the main limiting factors for performance and scalability are: a) the finite availability of resources such as registers and high-bandwidth/low-latency memory on computing nodes; b) the ability to compensate for any occurring latencies by having enough parallel tasks scheduled to prevent computing units from stalling or idling.

Consider for example the simulation of $1 \leq N \leq 10^7$ particles on a lattice with 18657 beam-elements, representing the LHC without beam-beam interactions. Evaluating SixTrackLib on a CPU-based OpenCL implementation (Intel Xeon E5-2630 20x2.2 GHz hyper-threads) and a range of high-end GPUs (NVIDIA Tesla V100 PCIe 16GB GPU) as well as consumer-grade GPUs (NVIDIA GTX 1050Ti 4GB, AMD RX560 4GB) demonstrates parallel speed-ups approaching (for large N) factors of 10^1 to and exceeding 10^2 (Fig. 4).

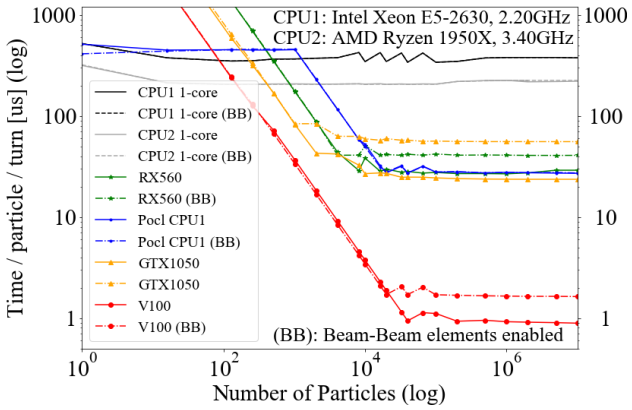

Figure 4: Benchmarking results of a LHC study using SixTrackLib on different target hardware. Increasing the complexity of the parallel code by enabling beam-beam elements (BB), but not using them, leads to decreased performance on GPUs, but not on CPU-based systems.

While enabling code-paths for handling beam-beam interactions in the parallel tracking code leads, unsurprisingly, to virtually no change on CPU-based systems, the increased complexity and pressure on resources impairs the performance on all studied GPU systems (cf. results in Fig. 4 labelled with (BB)). On lattices actually featuring complex features like beam-beam elements, this observation motivates studying ways to split the monolithic parallel code into smaller specialised blocks and to execute these blocks in sequence, thereby trading in synchronisation and dispatching overheads for a potentially better utilisation of hardware resources.

CONCLUSIONS

The SixTrack tracking code is the main code used to simulate long-term stability, collimation cleaning, and machine failure scenarios in the LHC, SPS and FCC due its unique features of speed and integration with HPC resources. It comes with a fully developed running environment to perform easily the massive numerical simulations that include scans on beam and ring parameters and the option of using different computing resources, from standard batch services to volunteer computing.

In spite of its maturity, SixTrack is still in an intense development phase. On the short time scale, it is planned to merge into a single code the features that were developed in the framework of the studies of the LHC collimation system. On a longer time scale, the main lines of development include tighter integration of existing features, interoperability with other codes, and deployment on new architectures such as GPU.

ACKNOWLEDGEMENTS

We wish to thank all volunteers that supported and continue to support the SixTrack project and the related studies by donating to the LHC@Home Project their CPU power and technical advice. We also wish to thank F. Schmidt for his continuous interest in SixTrack development.

REFERENCES

[1] F. Schmidt *et al.*, "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] SixTrack Project website, `http://cern.ch/sixtrack`

[3] R. Bruce *et al.*, "Status of Sixtrack with collimation", in *Proc. Tracking for Collimation Workshop*, CERN, Geneva, Switzerland, 2018.

[4] G. Demolaize *et al.*, "A New Version of SixTrack with Collimation and Aperture interfaces", in *Proc. PAC'05*, Knoxville, TN, USA, May 2005, paper FPAT081.

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

[6] L. Deniau *et al.*, "Upgrade of MAD-X for HL-LHC project and FCC studies", presented at ICAP'18, Key West, FL, USA, Oct 2018, paper TUPAF01, this conference.

[7] J. Barranco *et al.*, "LHC@Home: a BOINC-based volunteer computing infrastructure for physics studies at CERN", *Open Engineering*, vol. 7, pp. 378, 2017.

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

[9] K. Heinemann, G. Ripken, and F. Schmidt, "Construction of nonlinear symplectic six-dimensional thin-lens maps by exponentiation", DESY 95-189, 1995.

[10] D.P. Barber *et al.*, "A non-linear canonical formalism for the coupled synchro-betatron motion of protons with arbitrary energy", DESY 87-36, 1987.

[11] P. Hermes *et al.*, "Symplectic Tracking of Multi-Isotopic Heavy-Ion Beams in SixTrack", in *Proc. IPAC'16*, Busan, Korea, May 2016. `doi:10.18429/JACoW-IPAC2016-TUPMW015`

[12] M. Fjellstrom, "Particle Tracking in Circular Accelerators Using the Exact Hamiltonian in SixTrack", CERN-THESIS-2013-248, 2013.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAF02

[13] A. Latina and R. De Maria, "RF multipole implementation", CERN-ATS-Note-2012-088 TECH, 2012.

[14] L.H.A. Leunissen *et al.*, "6D beam-beam kick including coupled motion", *Phys. Rev. ST Accel. Beams*, vol. 3, pp. 124002, 2000.

[15] A. Patapenka, "First order transport map of wire element", in *3rd General SixTrack Meeting*, 7 December 2016, https://indico.cern.ch/event/590257/.

[16] V. Previtali *et al.*, "Numerical Simulations of a Hollow Electron Lens as a Scraping Device for the LHC", in *Proc. IPAC'13*, Shanghai, China, May 2013, paper MOPWO044.

[17] M. Fitterer *et al.*, "Implementation of Hollow Electron Lenses in SixTrack and first simulation results for the LHC", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017. doi:10.18429/JACoW-IPAC2017-THPAB041

[18] T. Trenkler and J. B. Jeanneret, "K2, a software package evaluating collimation systems in circular colliders (manual)", CERN SL/94105 (AP), 1994.

[19] C. Tambasco, "An improved scattering routine for collimation tracking studies at LHC", CERN-THESIS-2014-014, 2014.

[20] E. Quaranta, "Investigation of collimator materials for the High Luminosity Large Hadron Collider", CERN-THESIS-2017-101, 2017.

[21] Merlin source code repository, https://github.com/Merlin-Collaboration/Merlin

[22] Geant website, https://geant4.web.cern.ch/.

[23] K. Sjobak *et al.*, "New features of the 2017 SixTrack release", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017. doi:10.18429/JACoW-IPAC2017-THPAB047

[24] Pythia8 website, http://home.thep.lu.se/Pythia/.

[25] M. Berz, "The new method of TPSA Algebra for the description of beam dynamics to high orders", LANL Tech. Rep. AT-6:ATN-86-16, 1986.

[26] H. Mais and G. Ripken, "Theory of coupled synchro-betatron oscillations", DESY-M-82-05, DESY, 1982.

[27] K. Sjobak *et al.*, "General functionality for turn-dependent element properties in SixTrack", in *Proc. IPAC'15*, Richmond, USA, May 2015. doi:10.18429/JACoW-IPAC2015-MOPJE069

[28] K. Sjobak *et al.*, "Dynamic simulations in Sixtrack", in *Proc. Tracking for Collimation Workshop*, CERN, Geneva, Switzerland, 2018.

[29] L. Bottura, L Walckiers, and R. Wolf, "Field Errors Decay and "Snap-Back" in LHC Model Dipoles", CERN-LHC-Project-Report-55, 1996.

[30] A. Santamaría García *et al.*, "Limits on failure scenarios for crab cavities in the HL-LHC", in *Proc. IPAC'15*, Richmond, USA, May 2015. doi:10.18429/JACoW-IPAC2015-THPF095

[31] K. Sjobak, R. Bruce, H. Burkhardt, A. MacPherson, A. Santamaría García, and R. Kwee-Hinzmann, "Time Scale of Crab Cavity Failures Relevant for High Luminosity LHC", in *Proc. IPAC'16*, Busan, Korea, May 2016. doi:10.18429/JACoW-IPAC2016-THPOY043

[32] A. Santamaría García *et al.*, "Machine protection from fast crab cavity failures in the High Luminosity LHC", in *Proc. IPAC'16*, Busan, Korea, May 2016. doi:10.18429/JACoW-IPAC2016-TUPMW025

[33] S.J. Wretborn *et al.*, "Study of off-momentum losses at the start of the ramp in the Large Hadron Collider", CERN-ACC-NOTE-2017-0065, 2017.

[34] M. Fitterer *et al.*, "Implementation of hollow electron lenses in SixTrack and first simulation results for the HL-LHC", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017. doi:10.18429/JACoW-IPAC2017-THPAB041

[35] F. Schmidt, F. Willeke, and F. Zimmermann, "Comparison of methods to determine long-term stability in proton storage rings", CERN-SL-91-14-AP, 1991.

[36] M. Böoge and F. Schmidt, "Estimates for Long–Term Stability for the LHC", LHC Project Report 114, 1997.

[37] M. Giovannozzi *et al.*, "PLATO: a program library for the analysis of nonlinear betatronic motion", *Nucl. Instrum. and Methods A*, vol. 388, pp. 1, 1996.

[38] S. Kostoglou *et al.*, "Development of Computational Tools for Noise Studies in the LHC", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017. doi:10.18429/JACoW-IPAC2017-THPAB044

[39] J. Laskar, C. Froeschle, and C. Celletti, "The measure of chaos by the numerical analysis of the fundamental frequencies. Application to the standard mapping", *Physica D*, vol. 56, pp. 253, 1992.

[40] HDF5 website, https://www.hdfgroup.org/HDF5/.

[41] ROOT website, https://root.cern.ch/.

[42] V. Vlachoudis *et al.*, "Status of Fluka coupling to SixTrack", in *Proc. Tracking for collimation Workshop*, CERN, Geneva, Switzerland, 2018.

[43] T.T. Bohlen *et al.*, "The FLUKA Code: Developments and Challenges for High Energy and Medical Applications", *Nuclear Data Sheets*, vol. 120, 2014.

[44] A. Ferrari, P.R. Sala, A. Fassò, and J. Ranft, "FLUKA: a multi-particle transport code", CERN-2005-10, INFN/TC_05/11, SLAC-R-773, 2005.

[45] K. Sjobak *et al.*, "New Features of the 2017 SixTrack release", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017. doi:10.18429/JACoW-IPAC2017-THPAB047

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

[47] CERN Batch service, http://information-technology.web.cern.ch/services/batch

[48] High Throughput Condor (HTCondor), https://research.cs.wisc.edu/htcondor

[49] SixTrackLib source code repository, http://github.com/SixTrack/SixTrackLib

[50] J. Stone *et al.*, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems", *Computing in Science & Engineering*, vol. 12, pp. 3, 2010.

[51] J. Nickolls *et al.*, "Scalable Parallel Programming with CUDA", *ACM Queue*, vol. 6, pp. 2, 2008.