SixTrack V and runtime environment*

R. De Maria†

Beam Department (BE-ABP-HSS), CERN, 1211, Geneva 23, Switzerland

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, 1211, Geneva 23, Switzerland

E.H. Maclean

CERN, 1211, Geneva 23, Switzerland and
University of Malta, Msida, MSD 2080, Malta

K.N. Sjobak

CERN, 1211, Geneva 23, Switzerland and
University of Oslo, Boks 1072 Blindern, 0316, Oslo, Norway

I. Zacharov

EPFL, Rte de la Sorge, 1015, Lausanne, Switzerland

S. Singh

Indian Institute of Technology Madras, Chennai 600 036, India‡

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 external particle matter interaction codes. These features differentiate SixTrack from general-purpose, optics-design software. 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. This restructuring also removed a large number of compilation flags, instead enabling/disabling the functionality with run-time options. 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,

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

Radio-Frequency (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 several cluster environments available at CERN as well as submitting jobs to the LHC@Home volunteering computing project, which enables volunteers contributing with their hardware to CERN simulation. 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 library is able to run in CPUs as well as GPUs (graphical processing units). This contribution presents the status of the code, summarises the main existing features, and provides details about the main development lines SixTrack, SixDesk, and SixTrackLib.

1. Introduction

SixTrack[1] (available at `http://cern.ch/sixtrack`) 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.[2,3] 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.[2,3]

Different from a general-purpose code like MAD-X,[4] SixTrack is optimised for speed and numerical reproducibility. It can be also linked with the BOINC library[5] to use the volunteering computing project LHC@Home.[6] 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 [a], or LSF [b], or BOINC), 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.

2. Main features of SixTrack

SixTrack tracks an ensemble of particles defined by a set of coordinates through several beam-line elements, using symplectic maps,[7–9] or scattering elements.

[a]High Throughput Condor (HTCondor), `https://research.cs.wisc.edu/htcondor`
[b]IBM Spectrum LSF

2.1. *Coordinates*

The set of coordinates is larger than the minimum needed to describe the motion.

Table 1. Coordinates of the particles tracked by SixTrack. Extra dependent coordinates are stored to save expensive computations. Variables m_0, β_0, P_0 refer to the reference particle. SixTrack maps are symplectic when re-expressed in canonical coordinates.

x	Horizontal position
$x_p = \frac{P_x}{P}$	Horizontal direction cosine
y	Vertical position
$y_p = \frac{P_y}{P}$	Vertical direction cosine
$\sigma = s - \beta_0 ct$	Negative time delay w.r.t. the reference particle
$\delta = \frac{P\frac{m_0}{m} - P_0}{P_0}$	Relative momentum deviation w.r.t. the reference particle
E	Energy of the particle
P	Momentum of the particle
m	Mass of the particle
$\chi = \frac{q}{q_0}\frac{m_0}{m}$	Relative mass to charge ratio w.r.t. the reference particle
$r_v = \frac{\beta}{\beta_0}$	Velocity ratio w.r.t. the reference particle
$r_1 = \frac{1}{1+\delta}$	Quantity used in tracking maps
$r_2 = \frac{\chi}{1+\delta}$	Quantity used in tracking maps
$r_3 = \frac{1-\chi}{1+\delta}$	Quantity used in tracking maps
$r_4 = \frac{\delta}{1+\delta}$	Quantity used in tracking maps

Additional variables (see Table 1) are used to store energy-related quantities used in the tracking maps that are updated only on energy changes, which does not occur very frequently in synchrotrons in absence of radiation effects, to save computational time. Maps for dipoles and quadrupoles with non-zero length 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 also tracked at the same time[10] by correctly taking into account their relativistic speed as well as transverse magnetic rigidity. Variables used internally in tracking are not canonical, however, once they are converted to canonical form, the maps are symplectic. Differently from other codes such as MAD-X or PTC, SixTrack uses

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

as the longitudinal coordinate during tracking to avoid rounding errors associated to the relativistic β when updating σ in drifts [c].

2.2. *Beam-line elements*

Table 2 shows the different types of beam-line elements implemented in SixTrack. Thin multipoles are used in conjunction with the `MAKETHIN` and `SIXTRACK` com-

[c] $\Delta\sigma/l = 1 - \frac{\beta_0}{\beta}\left(1 + \frac{x_p^2+y_p^2}{2}\right)$ compared to $\Delta ct/l = \frac{1}{\beta_0} - \frac{1}{\beta}\left(1 + \frac{x_p^2+y_p^2}{2}\right)$ in MAD-X or PTC

Table 2. Beam line elements implemented in SixTrack.

Expanded drift	Drift resulting from an approximated Hamiltonian to second order
Exact drift[11]	Drift using the exact equation of motion
Single thin multipole	Infinitely thin magnetic multipole
Thin multiple block	Combination of magnetic multipole components up to the 20th order
Thick dipole-quadrupole	Combined magnetic dipole and quadrupole of a finite length
Thin solenoid	Infinitely thin solenoid
Accelerating cavity	Infinitely thin accelerating cavity
RF-multipoles[12]	Multipolar kick with longitudinal position sinusoidal dependence
4D Beam-beam	Transverse-only beam-beam interaction
6D beam-beam[13]	Full 6D beam-beam interaction with crossing angle, hourglass effects.
Wire[14]	Infinitely thin wire carrying current
Hollow electron lens[15,16]	Interaction with hollow electron distribution

mands in MAD-X to implement symplectic integrators for elements of finite length. 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 [d].

2.3. *Scattering*

SixTrack is capable of simulating the basic scattering processes undergone by an ultra-relativistic proton in the multi-TeV range when passing through matter.[2,3] 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.[17] Other scattering models can be imported and made available in the SixTrack executable, such as that of Merlin[e][18] and Geant4.[19,20]

A new module (SCATTER) 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 the *TOTal cross section, Elastic scattering and diffraction dissociation at the LHC Measurement* (TOTEM[21]). Alternatively, scattering events can be generated on the fly by Pythia8,[22] in which case elastic and diffractive processes are supported.

2.4. *Optics calculations*

SixTrack contains matrix code for 5D optics calculation and a 6D tracking engine using Truncated Power Series Algebra library (TPSA[23–25]) for 6D optics calculation.

[d] J. Andersson, "Comparison of tracking codes.", in CERN openlab summer students lighting talks 2018, `http://cds.cern.ch/record/2634289`.
[e] available at `https://github.com/Merlin-Collaboration/Merlin`

The 6D tracking engine uses the following canonical variables .

$$\left(x, \quad p_x = \frac{P_x}{P_0}\right), \quad \left(y, \quad p_y = \frac{P_y}{P_0}\right), \quad \left(\zeta = \frac{\beta}{\beta_0}\sigma, \quad \delta = \frac{P - P_0}{P_0}\right)$$

as conjugate canonical variables in 6D optics calculations which use explicitly symplectic maps. A symplecticity check is routinely performed on 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 to define strong beam sizes in beam-beam elements. The 6D optics module has been recently improved by removing some unnecessary ultra-relativistic approximations which introduced small symplectic errors.

2.5. *Dynamic effects*

A general functionality for dynamically-changing simulation settings on a turn-by-turn basis has been implemented.[27] 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,[28] HL-LHC crab cavity failure scenarios,[29–31] studies of beam losses during energy ramp,[32] and hollow electron-lens modulation.[33] 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.

2.6. *Online aperture checking*

SixTrack is capable of performing a check during tracking (online) of whether tracked particles still fall within the mechanical acceptance of the machine or not. The check is activated by the presence of aperture markers in the accelerator lattice. All aperture types of relevance for LHC, HL-LHC and FCC can be handled, namely: RECTELLIPSE (i.e. the aperture identified by the intersection of a rectangle with an ellipse), RACETRACK (i.e. the aperture identified by a rectangle with quadrants of ellispes at the corners) and OCTAGON (i.e. an octagon with sides at multiples of 45 degrees). The code is capable of handling horizontal and vertical offsets of the aperture markers, as well as rotations about the longitudinal axis. The implementation comes with a smooth, linear interpolation between all supported aperture markers, such that the aperture is very well defined at any location along the ring.

A smooth linear interpolation of the aperture model of the machine is essential to the detailed identification of the actual loss point tracked particles hitting the mechanical aperture of the machine (back-tracking algorithm). The loss location is identified comparing particle trajectories with the aperture model of the machine by means of a bi-section method; the user can set the precision of the identification of

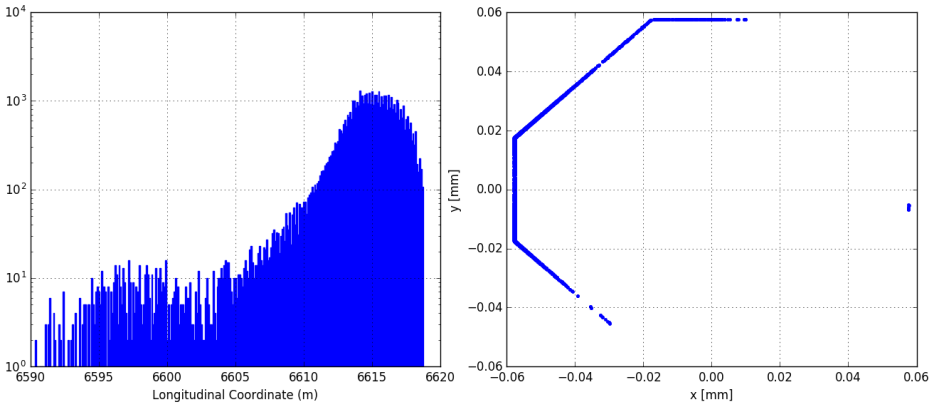

Figure 1. Longitudinal pattern (left frame) and transverse position (right frame) of losses in the inner triplet upstream of the IR5 experimental area for HL-LHC.[34] Losses get concentrated on the upper-left quadrant (right frame) because of the local position of the closed orbit. Simulation parameters: optics v1.3, β^*=15 cm, full non-linear model of the HL-LHC (no beam-beam collisions), no collimation system.

the longitudinal position. For the time being, the feature is available only for thin-lens tracking, where back-tracking is comfortably performed along a drift; extension to all other thick-lens linear elements is in plan. As an example, Figure 1 shows the longitudinal pattern and transverse position of losses in the inner triplet upstream of one experimental area for HL-LHC;[34] losses coming from dynamic aperture in absence of collimation system.

2.7. *Electron Lenses*

Electron lenses can be deployed to deplete over-populated beam tails in a controlled manner; as such, they are currently being investigated in the framework of the HL-LHC project.[34] The module is in under development. Recent changes include: porting to ion tracking, relevant for benchmarking recent measurements in RHIC[35] at Brookhaven National Laboratory; possibility to deploy measured radial profiles of electron beams; compatibility to the BOINC platform for volunteering computing, relevant for evaluating long-term effects on dynamic aperture. On-going development is focused on implementing 2D maps of measured electron profiles by means of Chebyshev polynomials.[36]

2.8. *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 2 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

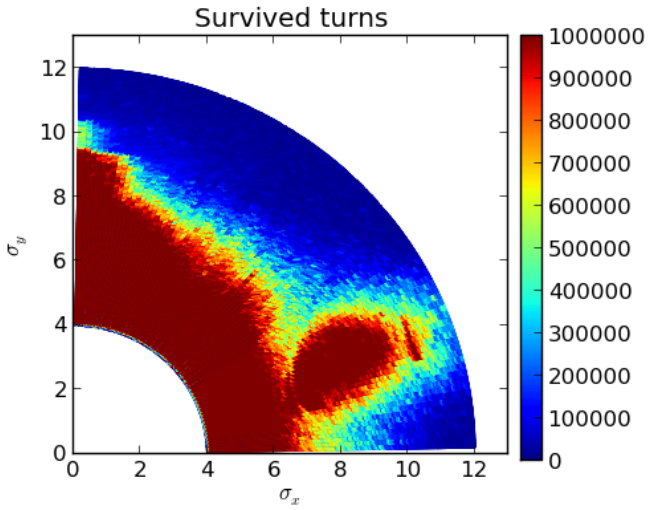

Figure 2. Survival plot of a fine phase-space scan for the LHC. The simulations was divided in task of 60 particles pairs covering the phase space in 2σ and 1.5° steps.

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 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.[37,38]

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.

2.9. *Frequency analysis*

A collection of routines for frequency analysis has been linked in SixTrack, namely PLATO[39] and a C++ implementation of the Numerical Analysis of Fundamental Frequency (NAFF) algorithm have been developed.[40] These algorithms allow for a more refined, compared to plain Fast-Fourier Transformations (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 computed.[41] With the resolution of the frequency map, many resonance lines become visible (example shown in Figure 3).

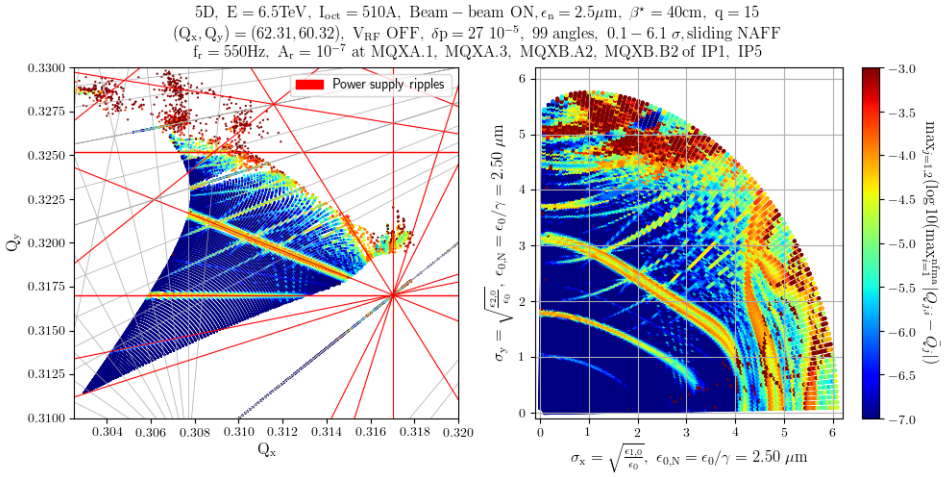

Figure 3. Frequency map using the NAFF method for LHC in the presence of a tune modulation in the triplets around the experimental areas with frequency of 550 Hz and relative amplitude of 10^{-7}.

2.10. *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 text (i.e. ASCII) and, in a few cases, a binary option is also available. Support for output of simulation data to Hierarchical Data For-

mat (HDF5[f]) files and ROOT[42] using the XrootD protocol 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.

2.11. *Interfaces to external programs*

Collimation studies can also be performed by running SixTrack coupled[43] to FLUKA,[44,45] a specialized simulation tool for describing the interaction (i.e. scattering) of particles with matter. 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. Coupling the two software tools in this manner efficiently combines the strengths of each program while, by virtue of using the same FLUKA geometries as for subsequent energy-deposition calculations run with FLUKA, also allows for an excellent level of consistency in the results.

Additionally, a more generalised interface "BDEX" (Beam Distribution EXchange) 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.[20]

2.12. *Building and testing*

A CMake-based build and test system has recently been added.[20] 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 lies in automating the execution of tests and the generation of pass/fail summary output. Full tests runs automatically every night and a subset of tests runs for each proposed code contribution (i.e. pull-request in git jargon).

2.13. *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

[f]available at `https://www.hdfgroup.org/HDF5/`

system memory and not by a build-time flag. A machine model like the LHC, using about 18k elements, including 4.6k high-order multipole blocks among them, needs about 220μs per particle per turn on a single CPU core at 3.4 GHz. Typical studies require in the order of $10^9 - 10^{12}$ particle turns or even more for parameter scans. For this reason, SixTrack is often used in conjunction with high-performance computing (HPC) facilities described in the following section.

3. SixDesk runtime environment

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 [g] as well as the BOINC platform for volunteering computing available at the LHC@Home project.[6] 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.

3.1. *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 intense SixTrack simulation campaigns (see Figure 4). 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 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,

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

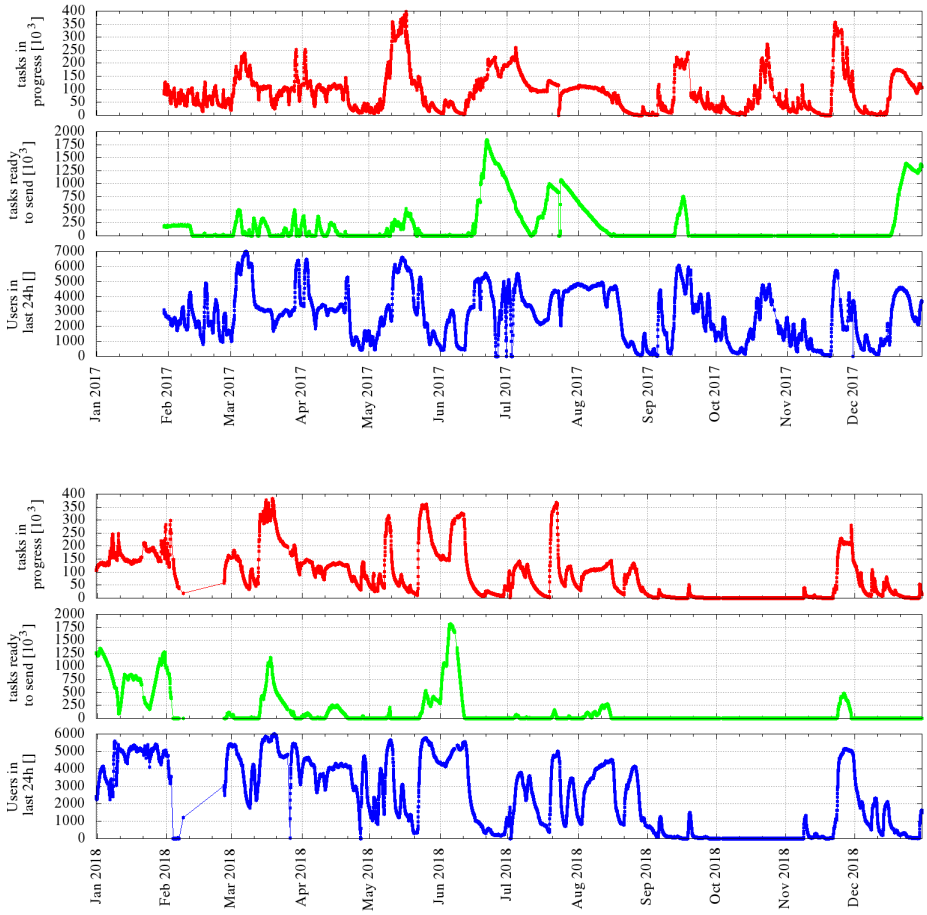

Figure 4. Summary of tasks and users during the last two years (upper frame: 2017; lower frame: 2018 to date) of the SixTrack application in the LHC@Home project. The number of users tends to increase, with the number of tasks absorbing most of the time all pending tasks.

contrary to LHC@Home, no redundancy is implemented during task submission since the code runs in a controlled environment, although very rarely hardware errors do appear in the results.

3.2. *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.

4. Main features of SixTrackLib

In the context of single-particle simulations, tracking requires no interaction between the calculations carried out for each particle. The memory requirements for representing each particle is about 160 Bytes. The machine description can, over a single turn, be considered constant, with different elements and sections of the ring requiring different amounts of local resources. SixTrack presents itself as an ideal candidate for a parallel implementation, because computations are strongly CPU bound with inherent parallelism and resource requirements do not scale 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 level 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,[6] 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 SixTrackLib [h], a clean-room implementation of the core functionality of SixTrack as a stand-alone library, allowing users to off-load the task of particle tracking onto supported HPC resources. The library provides a) a representation for a set of particles, b) a set of beam-elements such as drifts, magnetic multipoles (e.g. dipoles, quadrupoles, sextupoles, etc.), RF cavities, $4D$ and $6D$ beam-beam elements, etc. . . , c) a set of maps describing the tracking of particles over the beam-elements, d) a beam monitor lattice element, allowing to represent the functionality of beam instrumentation devices such as beam-position monitors, beam-profile monitors, etc., e) a range of contexts representing different parallelism and high-performance paradigms (e.g. auto-vectorized CPU code, OpenCL,[47] CUDA[48]) and the associated computing nodes within these paradigms, f) a dedicated generic buffer for managing collections of particles, beam elements and for transfer data to and from and the computing nodes, and g) a set of high-level APIs in C, C++ and Python.

The design of the library aims to completely separate the parts containing the physics (i.e. the particles, the beam elements, and the tracking maps) in a header-only module usable from C-like languages (e.g. C, C++, CUDA, OpenCL, etc.)

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

from the platform specific implementation pertinent to different computing environments and paradigms. Expressing the components in the abstracted header-only physics module allows the reuse of the modelled physics across all supported computing paradigms and to easily tap into the functionality of the tracking library from concrete and already existing applications and frameworks. Another benefit of this approach is the ability to extend the physics (for example by adding additional or alternative implementations for beam-elements) without the need to address different computing backends (or, vice versa, being able to add support for additional HPC environments without having to revisit the existing collection of physics related code-paths). Automatically generated code to aid users in such scenarios and to improve the maintainability of the code by reducing the library's footprint in terms of lines-of-code is currently under development.

SixTrackLib is developed in concert with pysixtrack [i], a reference implementation of the SixTrackLib physics using pure Python. It focuses on a straight forward and easy to understand version of the modelled physics, numerical reproducibility and numerical precision (by virtue of allowing the use of extended precision numerical types present in python) rather than performance or scalability. By comparing results between pysixtrack, SixTrackLib and SixTrack, the feasibility of the chosen design, the numerical accuracy and the performance can be evaluated and quality assured as part of the test-driven development efforts.

Consider for example a simulation on a lattice representing the LHC over a sequence of 100 turns while comparing the results generated via SixTrackLib against pysixtrack (Figure 5) and SixTrack (Figure 6). The presented magnitude of the absolute numerical difference in each of the six degrees of freedom is within admissible limits to prove the feasibility of the chosen design of both SixTrackLib and pysixtrack and lays the ground-work for further investigations and optimizations into this aspect.

Concerning an evaluation of the performance improvements achievable by using the parallel hardware capabilities of CPUs and GPUs via SixTrackLib and given the inherent parallelism of the problem outlined above, the main limiting factors for achieving high scalability and numerical throughputs are expected to be a) the finite availability of resources such as registers and high-bandwidth/low-latency memory on computing nodes, and b) the ability to compensate for any occurring latencies by having enough parallel tasks scheduled to prevent computing units from stalling or idling.

In order to study the impact of these effects, it is illustrative to simulate the same lattice using different transfer maps resulting in the same physics but requiring different amounts of hardware resources (e.g. registers). The by far most demanding transfer maps present in SixTrackLib as of this writing are the $4D$ and $6D$ beam-beam interaction implementations. By comparing the performance of simulating a lattice without beam-beam interactions once with a streamlined set of maps online

[i]pysixtrack source code repository, `https://github.com/rdemaria/pysixtrack`.

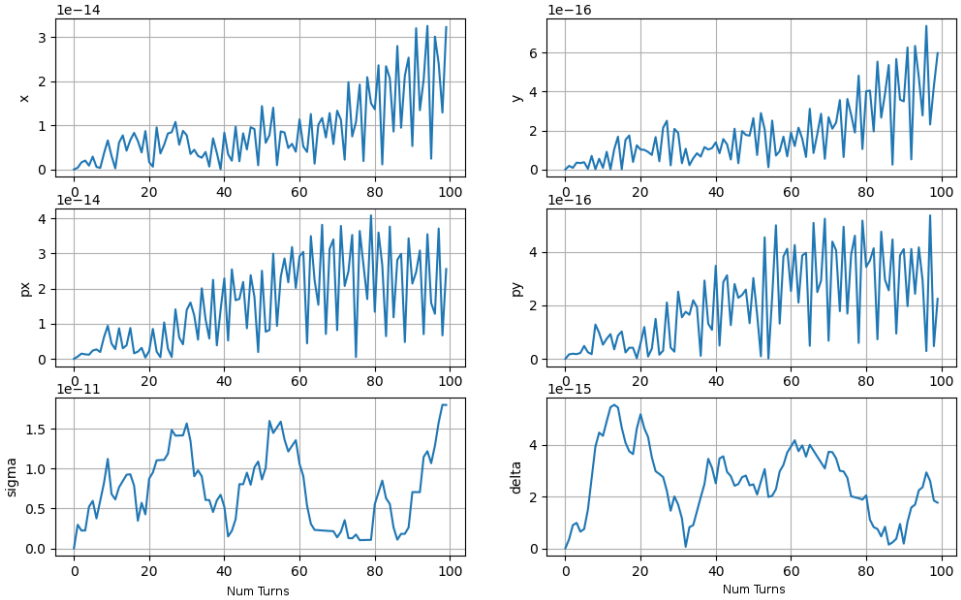

Figure 5. Absolute numerical difference of SixTrackLib and pysixtrack simulations of the LHC lattice over 100 turns.

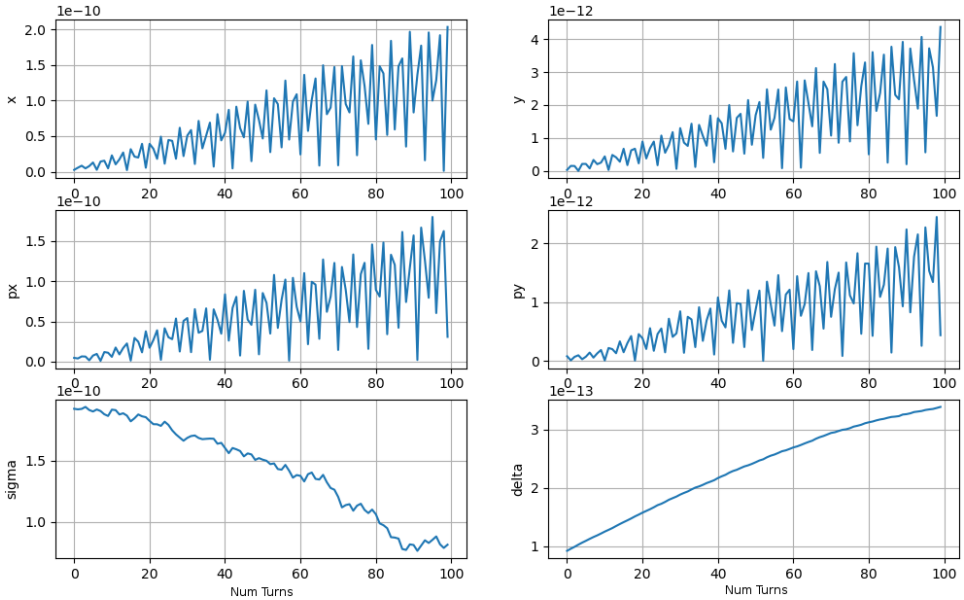

Figure 6. Absolute numerical difference of SixTrackLib and SixTrack simulations of the LHC lattice over 100 turns.

during the simulation and once with passive (i.e. unused) support for beam-beam interactions available, the effect can be illustrated and its impact in real-world scenarios estimated.

Consider to this end the simulation of $1 \leq N \leq 10^7$ particles on a lattice representing the LHC without beam-beam interactions using SixTrackLib, as presented in Figure 7. Evaluating the impact of the beam-beam maps using the OpenCL backend is especially interesting as it allows a direct comparison of the performance implications on both CPU and GPU configurations across vendors. The presented results encompass both CPUs (Intel Xeon E5-2630 20x2.2 GHz hyper-threads, AMD Ryzen 1950X, 16x3.40 GHz hyper-threads) and a range of both high-end GPUs (NVIDIA Tesla V100 PCIe 16GB GPU) as well as consumer-grade graphical processors (NVIDIA GTX 1050Ti 4GB, AMD RX560 4GB). The results demonstrate parallel speed-ups approaching (for large N) factors of 10^1 to and exceeding 10^2.

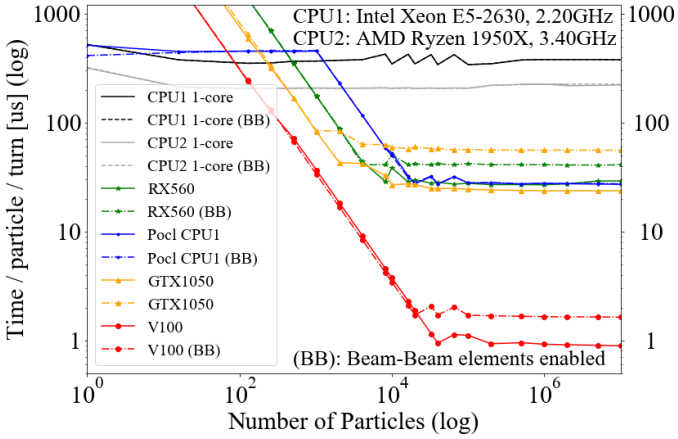

Figure 7. Benchmarking results of a LHC study using SixTrackLib on different target hardware. Increasing the complexity of the parallel code by enabling beam-beam elements (BB), but not using them, leads to decreased performance on GPUs, but not on CPU-based systems.

The impact of passively enabling the beam-beam interactions (labelled using (BB) in Figure 7) in the parallel tracking code leads, unsurprisingly, to virtually no change on CPU-based systems; these systems are not as sensitive to register pressure and resource starvation as graphical processor based environments. For the GPU based configurations, the increased complexity and pressure on resources impairs the performance on all studied systems, although the degree of the impact and the economics of keeping complex maps on-line even if the features are not actively used are differing considerably even within the context of this limited set of considered hardware configurations. The results motivate studying ways to split the monolithic parallel code into smaller specialised blocks and to provide APIs and facilities to execute these blocks in sequence, thereby trading in synchronisation and dispatching overheads for a potentially better utilisation of hardware resources.

5. 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 to its unique features of speed and integration with HPC resources. It comes with a fully developed running environment to easily perform 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 all the code and combine all the features developed within the framework of the studies of the LHC collimation system into a single code-base. 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 GPUs.

6. 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.

Bibliography

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. R. Bruce *et al.*, "Status of Sixtrack with collimation", in *Proc. Tracking for Collimation Workshop*, CERN, Geneva, Switzerland, 2018, pp. 1-10.
3. A. Mereghetti *et al.*, "SixTrack for cleaning studies: 2017 updates", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, paper THPAB046.
4. L. Deniau *et al.*, "Upgrade of MAD-X for HL-LHC project and FCC studies", in *Proc. ICAP'18*, Key West, FL, USA, Oct 2018, paper TUPAF01.
5. D. P. Anderson, "'BOINC: a System for Public-Resource Computing and Storage', in *Proc. of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, 2004, pp. 4-10.
6. J. Barranco *et al.*, "LHC@Home: a BOINC-based volunteer computing infrastructure for physics studies at CERN", *Open Engineering* **7** 378 (2017).
7. G. Ripken and F. Schmidt, "A symplectic six-dimensional thin-lens formalism for tracking", DESY 95-63, CERN/SL/95-12(AP), 1995.
8. K. Heinemann, G. Ripken, and F. Schmidt, "Construction of nonlinear symplectic six-dimensional thin-lens maps by exponentiation", DESY 95-189, 1995.
9. 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.
10. P. D. Hermes, "Heavy-Ion Collimation at the Large Hadron Collider - Simulations and Measurements", Ph. D. Thesis, University of Münster, Münster, Germany (2016).
11. M. Fjellstrom, "Particle Tracking in Circular Accelerators Using the Exact Hamiltonian in SixTrack", CERN-THESIS-2013-248, 2013.

12. A. Latina and R. De Maria, "RF multipole implementation", CERN-ATS-Note-2012-088 TECH, 2012.
13. L.H.A. Leunissen *et al.*, "6D beam-beam kick including coupled motion", *Phys. Rev. ST Accel. Beams* **3** 124002 (2000).
14. A. Patapenka *et al.*, "Simulation in support of wire beam-beam compensation experiment at the LHC", in *Proc. NAPAC'16*, Chicago, IL, USA, Oct 2016, paper TUPOB17.
15. 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.
16. 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, paper THPAB041.
17. E. Quaranta, "Investigation of collimator materials for the High Luminosity Large Hadron Collider", Ph. D. Thesis, Politecnico di Milano, Milan, Italy (2017).
18. S. Tygier *et al.*, "Simulating Novel Collimation Schemes for High-Luminosity LHC With Merlin++", in *Proc. IPAC'19*, Melbourne, Australia, May 2019, paper MOPRB060.
19. J. Allison *et al.*, "Recent developments in Geant4", in *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 835, 2016, Pages 186-225.*
20. K. Sjobak *et al.*, "New features of the 2017 SixTrack release", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, paper THPAB047.
21. G. Anelli *et al.* (TOTEM Collaboration) (2008). "The TOTEM Experiment at the CERN Large Hadron Collider". Journal of Instrumentation. 3 (8): S08007, doi:10.1088/1748-0221/3/08/S08007.
22. T. Sjstrand *et al.*, "A brief introduction to PYTHIA 8.1", in *Computer Physics Communications*, 11, 2008, Jun, 178, 852867
23. M. Berz, "The Method of Power Series Tracking for the Mathematical Description of Beam Dynamics", *Nucl. Instrum. and Methods A* **258** (1987) 431-436.
24. M. Berz, "Differential Algebraic Description of Beam Dynamics to Very High Orders", *Particle Accelerators 24* (1989) 109-124.
25. M. Berz, "Modern Map Methods in Particle Beam Physics", Academic Press, 1999, ISBN 0-12-014750-5.
26. H. Mais and G. Ripken, "Theory of coupled synchro-betatron oscillations", DESY-M-82-05, DESY, 1982.
27. K. Sjobak *et al.*, "Dynamic simulations in Sixtrack", in *Proc. Tracking for Collimation Workshop*, CERN, Geneva, Switzerland, 2018, pp. 123-134.
28. L. Bottura, L Walckiers, and R. Wolf, "Field Errors Decay and "Snap-Back" in LHC Model Dipoles", CERN-LHC-Project-Report-55, 1996.
29. A. Santamaría García *et al.*, "Limits on failure scenarios for crab cavities in the HL-LHC", in *Proc. IPAC15*, Richmond, USA, May 2015, paper THPF095.
30. K. Sjobak *et al.*, "Time Scale of Crab Cavity Failures Relevant for High Luminosity LHC", in *Proc. IPAC16*, Busan, Korea, May 2016, paper THPOY043.
31. A. Santamaría García *et al.*, "Machine protection from fast crab cavity failures in the High Luminosity LHC", in *Proc. IPAC16*, Busan, Korea, May 2016, paper TUPMW025.
32. 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.
33. M. Fitterer *et al.*, "Implementation of hollow electron lenses in SixTrack and first simulation results for the HL-LHC", in *Proc. IPAC17*, Copenhagen, Denmark, May

2017, paper THPAB041.
34. G. Apollinari *et al.*, "High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1", CERN-2017-007-M, CERN, Geneva, Switzerland.
35. D. Mirarchi *et al.*, "Hollow electron-lenses assisted collimation and plans for the LHC", in *Proc. HB2018*, Daejeon, Korea, June 2018, paper TUP1WE02.
36. G. Stancari, "Calculation of the transverse kicks generated by the bends of a hollow electron lens", FERMILAB-FN-0972-APC, Fermilab, Batavia, Illinois, USA (2014).
37. 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.
38. M. Böoge and F. Schmidt, "Estimates for LongTerm Stability for the LHC", LHC Project Report 114, 1997.
39. M. Giovannozzi *et al.*, "PLATO: a program library for the analysis of nonlinear betatronic motion", *Nucl. Instrum. and Methods A* **388** 1 (1996).
40. S. Kostoglou *et al.*, "Development of Computational Tools for Noise Studies in the LHC", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, paper THPAB044.
41. 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* **56** 253 (1992).
42. R. Brun and F. Rademakers, "ROOT - An Object Oriented Data Analysis Framework", Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86.
43. E. Skordis *et al.*, "FLUKA coupling to SixTrack", in *Proc. Tracking for collimation Workshop*, CERN, Geneva, Switzerland, 2018, pp. 17-26.
44. G. Battistoni *et al.*, "Overview of the FLUKA code", *Annals of Nuclear Energy* **82**, 10–18 (2015).
45. 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.
46. E. McIntosh and R. De Maria, "The SixDesk Run Environment for SixTrack", CERN-ATS-Note-2012-089 TECH, 2012.
47. J. Stone *et al.*, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems", *Computing in Science & Engineering* **12** 3 (2010).
48. J. Nickolls *et al.*, "Scalable Parallel Programming with CUDA", *ACM Queue* **6** 2 (2008).