

An Integrated Beam Physics Simulation Framework

**G. Iadarola, R. De Maria, S. Łopaciuk,
A. Abramov, X. Buffat, D. Demetriadou, L. Deniau, P. Hermes, P. Kicsiny, P. Kruyt, A. Latina,
L. Mether, K. Paraschou, G. Sterbini, F. Van Der Veken, CERN, Geneva, Switzerland**

P. Belanger, TRIUMF, Vancouver, Canada

D. Di Croce, T. Pieloni, L. Van Riesen-Haupt, M. Seidel, EPFL, Lausanne, Switzerland

P. Niedermayer, GSI, Darmstadt, Germany

Work supported by:  CHART

The CHART logo is a red, irregular shape resembling a map of Switzerland, with several small white stars scattered across it.

<https://xsuite.web.cern.ch>



- **Introduction**
 - Motivation
 - Design goals and constraints
 - Architecture
 - Agile development
- **Lattice modeling, simulation and optimization**
 - Lattice modeling
 - Single-particle tracking
 - Dynamic parameter control
 - Multi-objective optimizer
- **Simulation of specific processes**
 - Particle-matter interaction
 - Synchrotron radiation
 - Handling of collective effects
 - Space-charge
 - Beam-beam
 - Electron clouds
- **Final remarks**



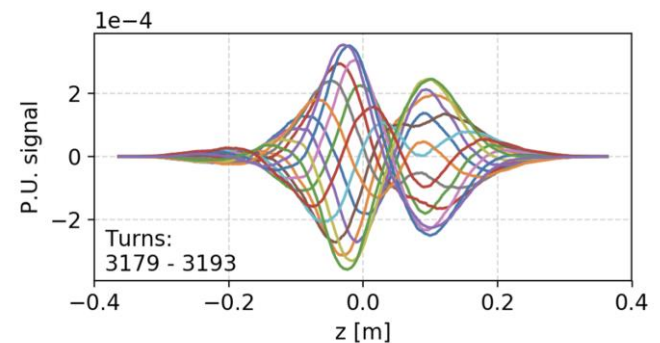
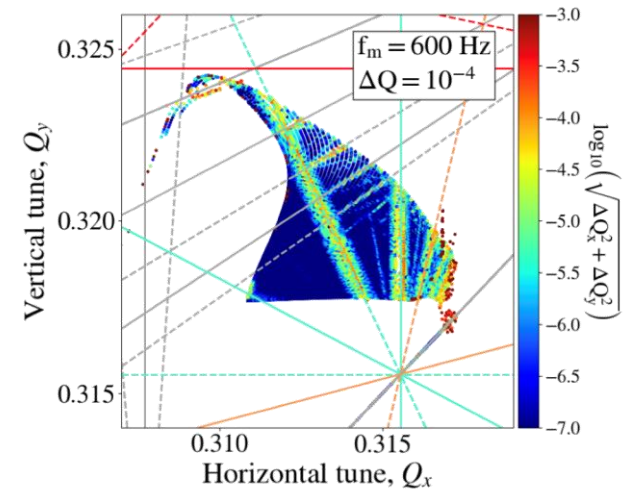
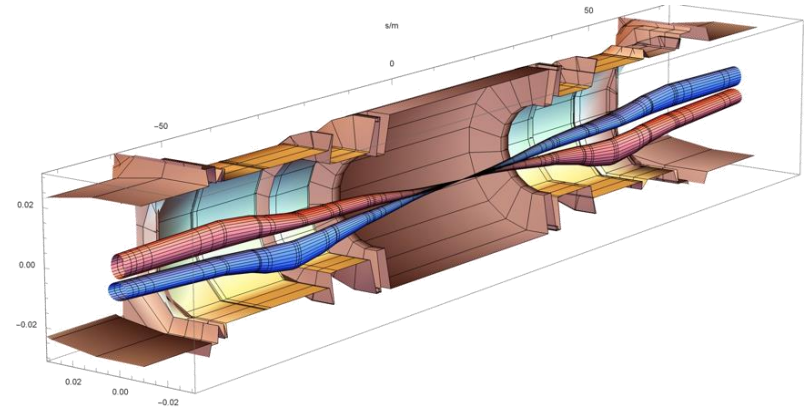
- **Introduction**
 - Motivation
 - Design goals and constraints
 - Architecture
 - Agile development
- **Lattice modeling, simulation and optimization**
 - Lattice modeling
 - Single-particle tracking
 - Dynamic parameter control
 - Multi-objective optimizer
- **Simulation of specific processes**
 - Particle-matter interaction
 - Synchrotron radiation
 - Handling of collective effects
 - Space-charge
 - Beam-beam
 - Electron clouds
- **Final remarks**

CERN has a **long tradition** in the **development of software tools for beam physics**

Powerful **tools** provided to the user community:

- **MAD-X**, standard for lattice description, optics calculation and design, tracking
- **Sixtrack**, a **fast-tracking program** used mainly for long **single-particle simulations**
- **Sixtracklib**, a **C/C++ library for single-particle tracking** compatible with Graphics Processing Units (GPUs)
- **COMBI**, for the simulation of **beam-beam** effects using **strong-strong modelling**
- **PyHEADTAIL**, a **Python** toolkit for **collective effects** (impedance, feedbacks, space charge, and e-cloud).

Developed over decades, providing **advanced features** in their respective domains



Over the years we started to **face needs that could not be easily fulfilled with legacy tools:**

- **Integration**: difficult to combine features from different codes to simulate complex heterogeneous effects
- **User interface**: need to move away from custom interfaces (text files / ad-hoc scripting) towards standard **Python packages**
 - Present **de-facto standard in scientific computing**
 - **Easy to combine** in notebooks and or in more complex Python codes
 - Allows **leveraging an ever-growing arsenal of general-purpose Python libraries** (statistics, linear algebra, frequency analysis, optimization, plot, etc.)
 - Boosted by **substantial investments from general industry** (big data, AI)
- **GPU acceleration**: mandatory for several applications but cumbersome to retrofit in the existing codes



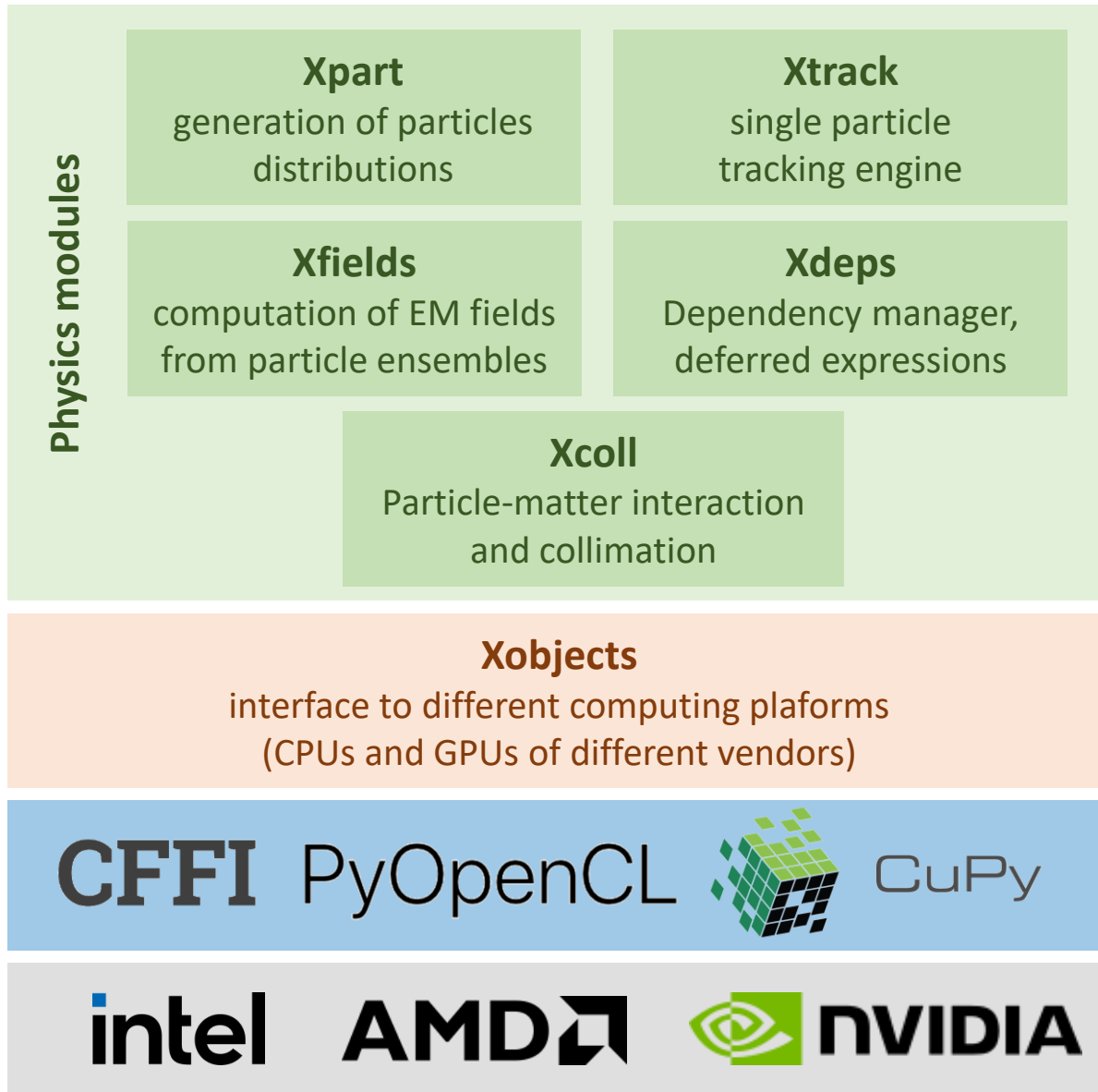
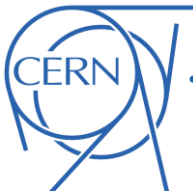


This led to the **launch of the Xsuite project** in **2021**

- **Main goal:** bring into a **modern Python toolkit** the know-how built in developing and exploiting MAD, Sixtrack, COMBI, PyHEADTAIL etc.,
- Designed for **seamless integration** among the different components and for **extendability**
- Designed to support **different computing platforms**, including **multicore CPUs and GPUs** from different vendors

Design constraints:

- Need to cover a **large spectrum of phenomena and applications** (flexibility!)
- Need **developer learning curve** to be short as possible
 - Features developed **directly by field experts**
- Need **to grow the code in a “sustainable” way**, being managed and maintained by a **small core team experts** integrating (in a clean way!) contributions by a wide developer community



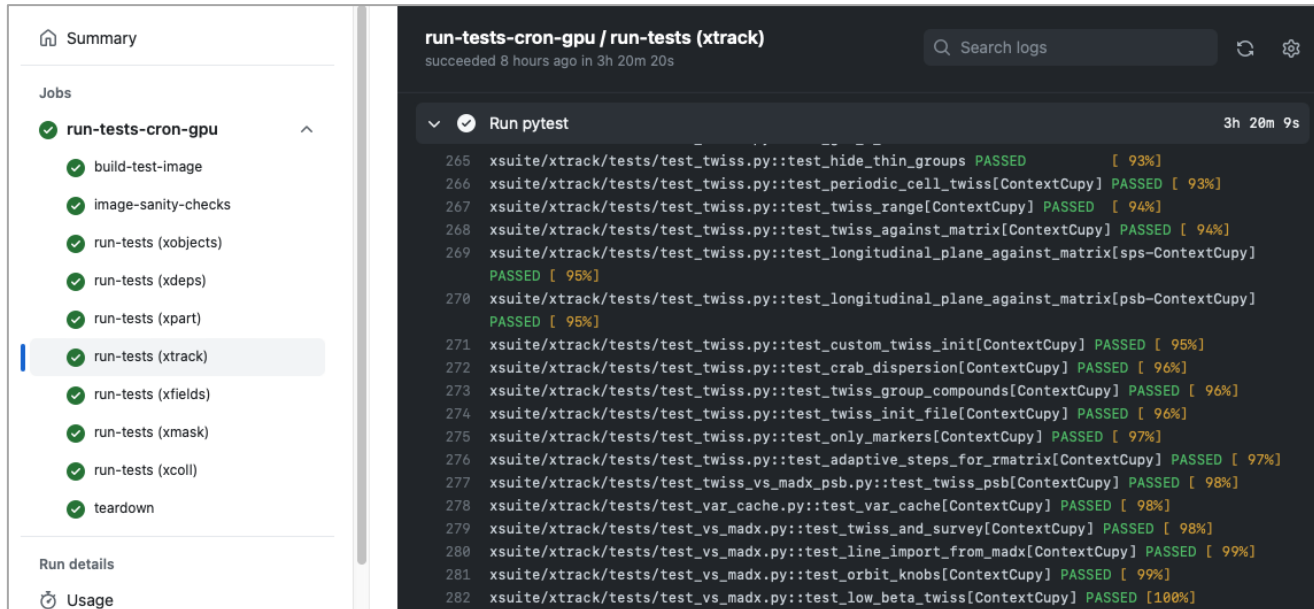
Project employs an “orthogonal” design strategy:

- Ensure that **each functional block** remains **well-isolated** and interacts with the others through **clearly defined interfaces**
- This approach has two key **advantages**:
 - Enables **contributors** to **modify or expand** specific components **without full knowledge of other parts** nor of underlying software infrastructure
 - **Minimize codebase complexity**, ensuring that it **increases linearly rather than exponentially** as new features are added



Our development follows the **agile principles**:

- Since early states **we engaged with the user community**, encouraging and **supporting users** to **test and exploit** available features in **full-scale studies**
- Code **evolves incrementally**, promptly applying needed fixes and improvements
- **Rely on fast release cycle** (new versions released multiple times per month) while ensuring **no disruption due to version changes** on the user's side.
- Made possible by large investment **automatic testing**: each version of Xsuite undergoing over a **thousand automatic checks** (on CPU and GPU)



The screenshot displays a CI/CD pipeline interface. On the left, a 'Summary' panel lists several jobs, all of which are marked with green checkmarks, indicating they passed. The jobs listed are: build-test-image, image-sanity-checks, run-tests (xobjects), run-tests (xdeps), run-tests (xpart), run-tests (xtrack), run-tests (xfields), run-tests (xmask), run-tests (xcoll), and teardown. The 'run-tests (xtrack)' job is currently selected. The main panel shows the detailed log for the 'Run pytest' step of the 'run-tests-cron-gpu / run-tests (xtrack)' pipeline. The log indicates that the step succeeded 8 hours ago in 3h 20m 20s. The log content shows a series of test results, all of which passed with varying percentages of success: 93%, 94%, 94%, 95%, 95%, 95%, 96%, 96%, 96%, 97%, 97%, 98%, 98%, 98%, 99%, 99%, and 100%.

```
run-tests-cron-gpu / run-tests (xtrack)
succeeded 8 hours ago in 3h 20m 20s

Run pytest 3h 20m 9s
265 xsuite/xtrack/tests/test_twiss.py::test_hide_thin_groups PASSED [ 93%]
266 xsuite/xtrack/tests/test_twiss.py::test_periodic_cell_twiss[ContextCupy] PASSED [ 93%]
267 xsuite/xtrack/tests/test_twiss.py::test_twiss_range[ContextCupy] PASSED [ 94%]
268 xsuite/xtrack/tests/test_twiss.py::test_twiss_against_matrix[ContextCupy] PASSED [ 94%]
269 xsuite/xtrack/tests/test_twiss.py::test_longitudinal_plane_against_matrix[sps-ContextCupy]
PASSED [ 95%]
270 xsuite/xtrack/tests/test_twiss.py::test_longitudinal_plane_against_matrix[psb-ContextCupy]
PASSED [ 95%]
271 xsuite/xtrack/tests/test_twiss.py::test_custom_twiss_init[ContextCupy] PASSED [ 95%]
272 xsuite/xtrack/tests/test_twiss.py::test_orab_dispersion[ContextCupy] PASSED [ 96%]
273 xsuite/xtrack/tests/test_twiss.py::test_twiss_group_compounds[ContextCupy] PASSED [ 96%]
274 xsuite/xtrack/tests/test_twiss.py::test_twiss_init_file[ContextCupy] PASSED [ 96%]
275 xsuite/xtrack/tests/test_twiss.py::test_only_markers[ContextCupy] PASSED [ 97%]
276 xsuite/xtrack/tests/test_twiss.py::test_adaptive_steps_for_rmatrix[ContextCupy] PASSED [ 97%]
277 xsuite/xtrack/tests/test_twiss_vs_madx_psb.py::test_twiss_psb[ContextCupy] PASSED [ 98%]
278 xsuite/xtrack/tests/test_var_cache.py::test_var_cache[ContextCupy] PASSED [ 98%]
279 xsuite/xtrack/tests/test_vs_madx.py::test_twiss_and_survey[ContextCupy] PASSED [ 98%]
280 xsuite/xtrack/tests/test_vs_madx.py::test_line_import_from_madx[ContextCupy] PASSED [ 99%]
281 xsuite/xtrack/tests/test_vs_madx.py::test_orbit_knobs[ContextCupy] PASSED [ 99%]
282 xsuite/xtrack/tests/test_vs_madx.py::test_low_beta_twiss[ContextCupy] PASSED [100%]
```



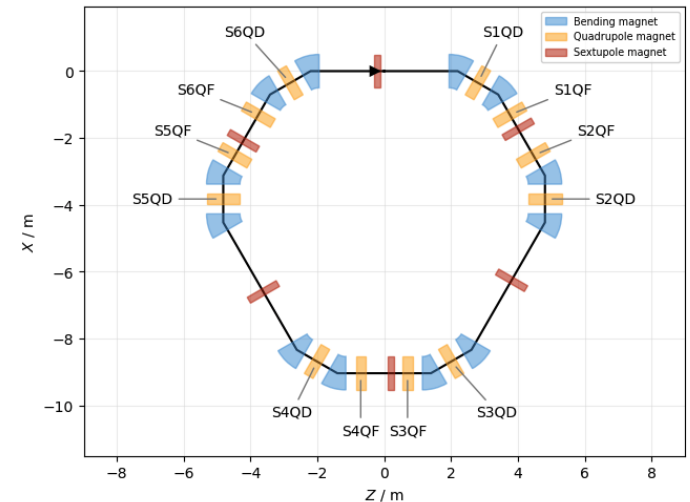
- **Introduction**
 - Motivation
 - Design goals and constraints
 - Architecture
 - Agile development
- **Lattice modeling, simulation and optimization**
 - Lattice modeling
 - Single-particle tracking
 - Dynamic parameter control
 - Multi-objective optimizer
- **Simulation of specific processes**
 - Particle-matter interaction
 - Synchrotron radiation
 - Handling of collective effects
 - Space-charge
 - Beam-beam
 - Electron clouds
- **Final remarks**

- The **beam line** is represented as a **sequence of Python objects**, each corresponding to an accelerator element or to other physical processes (e.g. magnets, cavities, aperture restrictions, etc.).
 - Can be **defined manually** or **imported from MAD-X**
 - Including **tilts, misalignments** and **multipolar errors**

Xsuite model of a ring
(represented with the [Xplt package](#))

```
survey = tracker.survey()
```

```
plot = xplt.FloorPlot(survey, line, labels="S.Q.")
plot.legend()
```



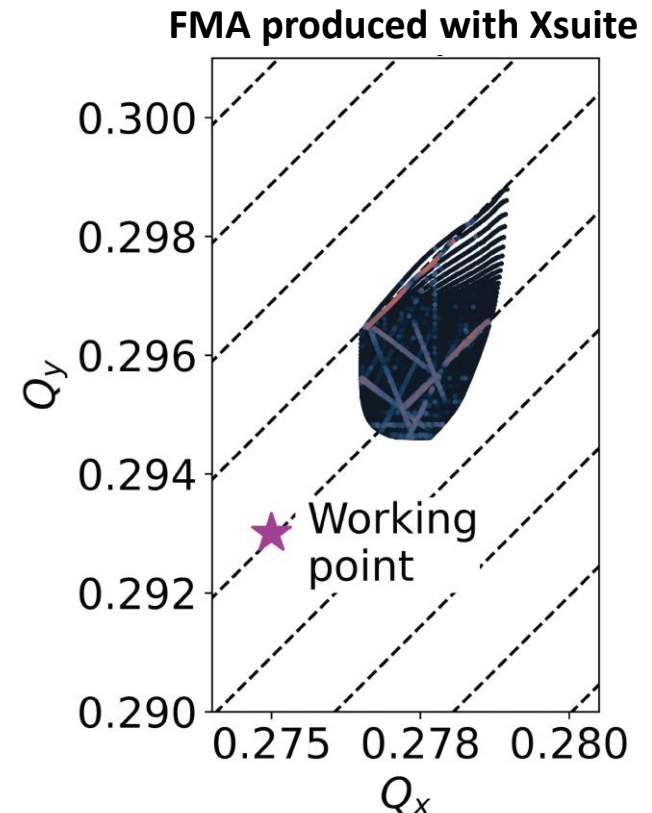
We provide:

- **“Thin” lattice integration**, largely based on the Sixtrack and Sixtracklib experience
- **“Thick” maps** for bending and quadrupole magnets are also available:
 - For bends, we provide a **“full” map** (appropriate for small rings), and an **“expanded” map** (for large rings and small bending angles).
- **Dipole edge effects** including **fringe fields** can be modeled either in their **linearized form** or as **full non-linear maps** (same fringe model as in MAD-NG and PTC).

- To **speed up tracking simulations**, Xsuite assembles and compiles a **C kernel** (callable from Python) **optimized for the given beamline** and **specialized for the chosen platform** (CPU or GPU)
 - The **tracking speed** is found to be **similar to Sixtrack** for single-core CPU and **about two orders of magnitudes faster than that on high-end GPUs**
 - Developments are well advanced to deploy Xsuite on the **LHC@Home** volunteer computing platform

Tracking time for a typical LHC simulation

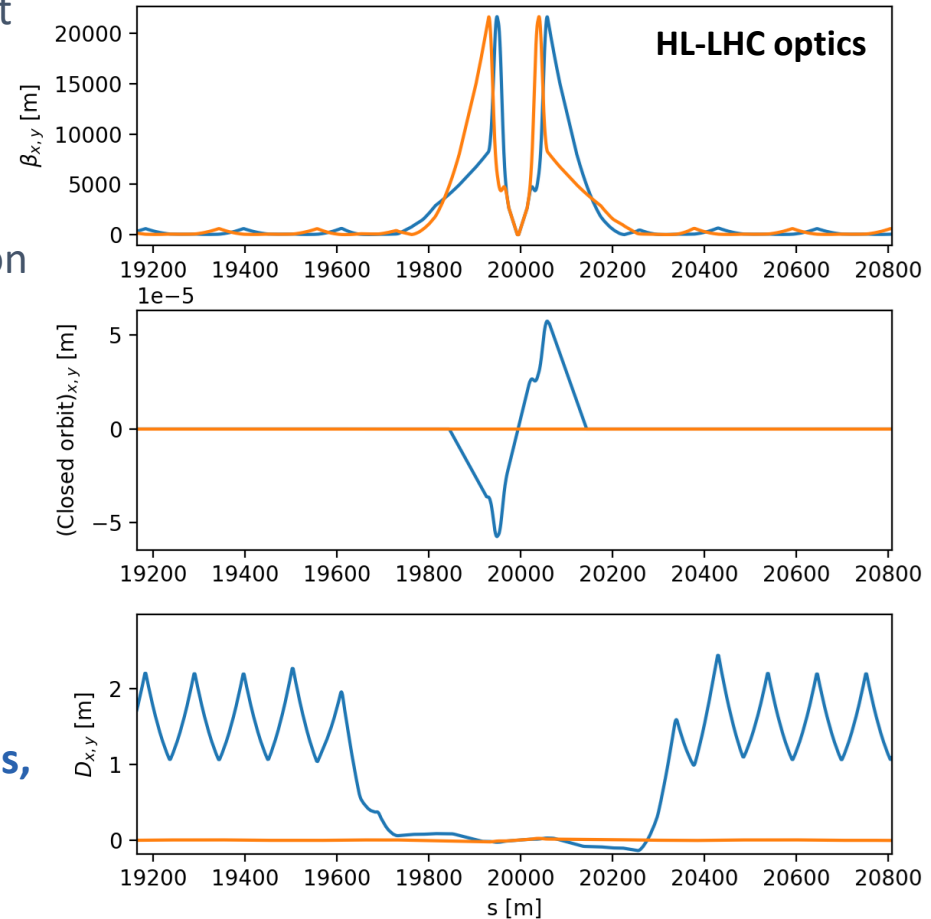
Platform	Computing time
CPU (single core)	190 ($\mu\text{s}/\text{part.}/\text{turn}$)
GPU (NVIDIA V100, cupy)	0.80 ($\mu\text{s}/\text{part.}/\text{turn}$)
GPU (NVIDIA V100 pyopencl)	0.85 ($\mu\text{s}/\text{part.}/\text{turn}$)





$$q_x = 0.31000 \quad q_y = 0.32000$$
$$Q'_x = 2.00 \quad Q'_y = 2.00 \quad \gamma_{tr} = 53.57$$

- The **Xsuite Twiss** module can be used to extract **lattice functions** of a ring or a beamline
- The calculation **probes the lattice simply by tracking particles**:
 - **Closed orbit** obtained by applying a Python root finder on the tracking
 - The **Jacobian** matrix obtained by **tracking (central differences)**
 - Compute “**Linear Normal Form**” of the Jacobian matrix (diagonalization)
 - **Propagate eigenvectors** by **tracking**
 - Obtain from the eigenvectors **Twiss parameters** (α, β, γ), **dispersion functions**, **phase advances**, **coupling coefficients**
- Computation can be done with **assigned beam momentum** to get **off-momentum beta-beating**, **non-linear chromaticity** etc.



Accuracy compared to MAD-X: $\Delta\beta / \beta \ll 10^{-4}$

Computation time is very similar

```
In [37]: tw.bety[0] # xsuite
Out[37]: 149.4305507849305

In [38]: mad.table.twiss.bety[0] # madx
Out[38]: 149.43055000962505

In [39]: t_mad_ms
Out[39]: 202.0

In [40]: t_xsuite_ms
Out[40]: 185.0
```

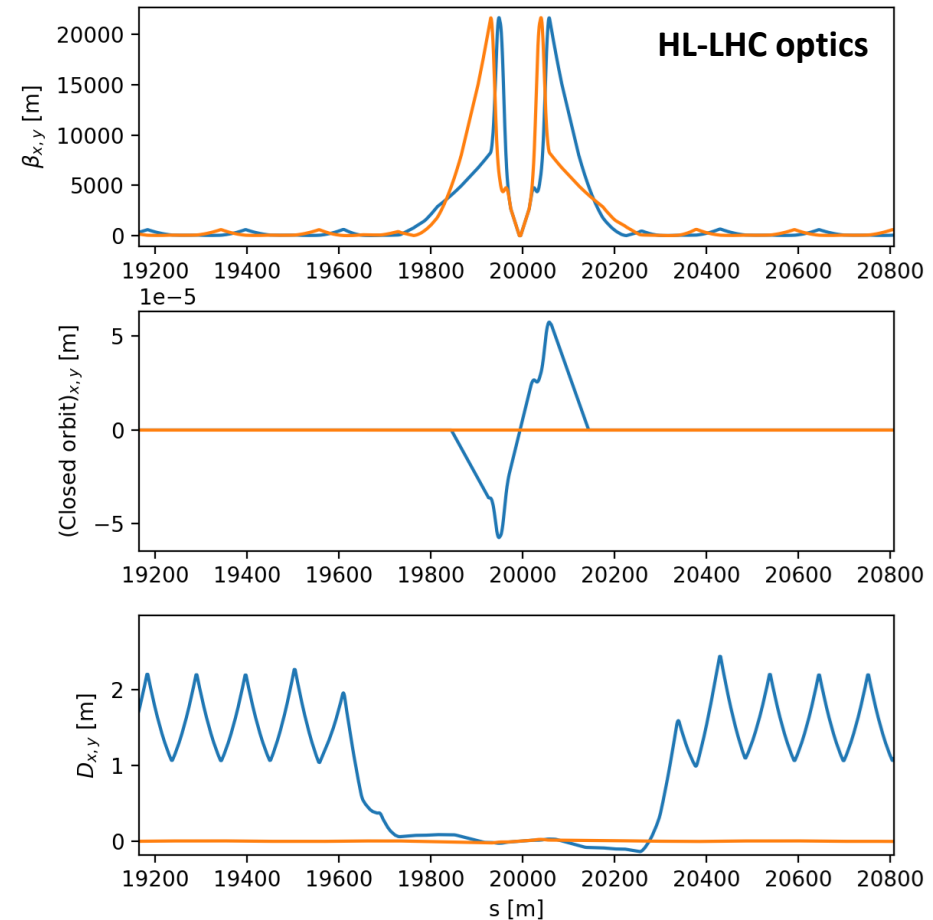
Example

$$q_x = 0.31000 \quad q_y = 0.32000$$

$$Q'_x = 2.00 \quad Q'_y = 2.00 \quad \gamma_{tr} = 53.57$$

Computation of **Twiss parameters based on the tracking** has **two main advantages**:

- Any **physical model included in the tracking** is **automatically usable in Twiss**
 - Without additional development effort
- Twiss becomes a **powerful diagnostics tool** on the built **tracking model**
 - Allows **measuring directly on the tracking model** tunes, chromaticities, closed orbit, beta functions, etc.
 - Can be done **effortlessly** and **without exporting or manipulating the model**.
 - **Used daily** to for validating simulation models, catching mistakes, investigating issues



Accuracy compared to MAD-X: $\Delta\beta / \beta \ll 10^{-4}$
Computation time is very similar

```
In [37]: tw.bety[0] # xsuite
Out[37]: 149.4305507849305

In [38]: mad.table.twiss.bety[0] # madx
Out[38]: 149.43055000962505

In [39]: t_mad_ms
Out[39]: 202.0

In [40]: t_xsuite_ms
Out[40]: 185.0
```

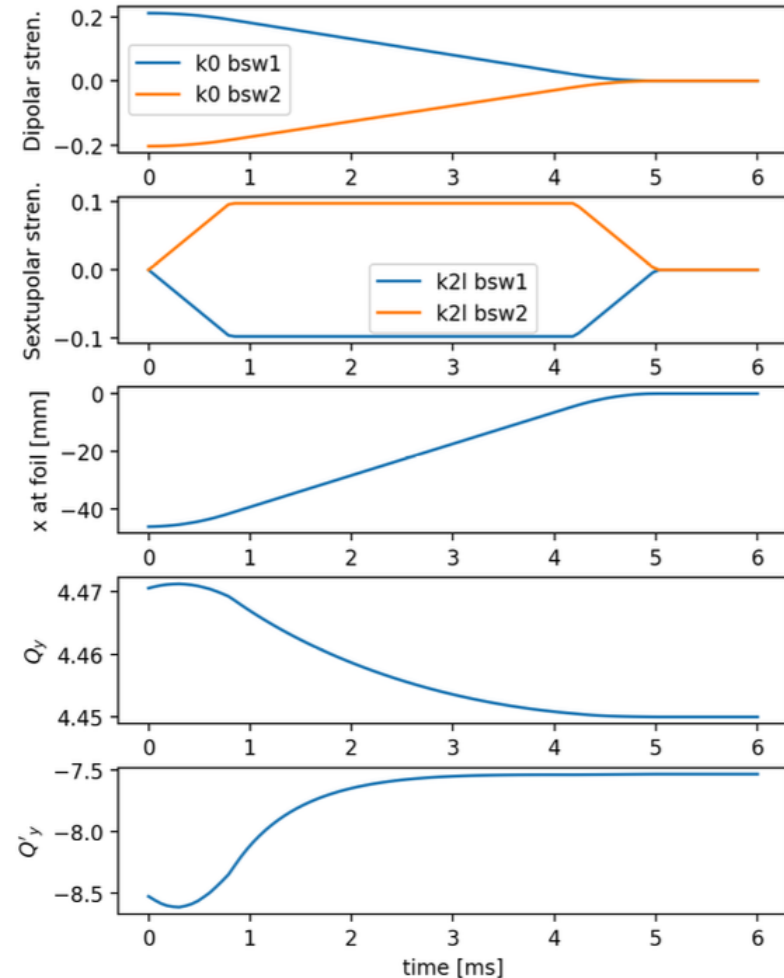
Example

- In accelerators often a single **high-level parameter** can be used to control **groups of components** with complex dependency relations. (e.g. magnets in series, groups of RF cavities, etc.)
- The **Xdeps module** provides the capability to include such dependencies in the simulation model (as done by MAD-X deferred expressions)
- Example**, LHC crossing angle knob:
 - At any time, the user can set:


```
lhc.vars['on_x1'] = 160 # murad
```

 which **automatically changes the strength of 40 dipole correctors** to get the required crossing angle
- User can also use **“Time functions”**, i.e. **time dependent knobs** that are updated automatically during the simulation

Simulation of a fast orbit bump used for the H⁻ injection into the CERN PS Booster



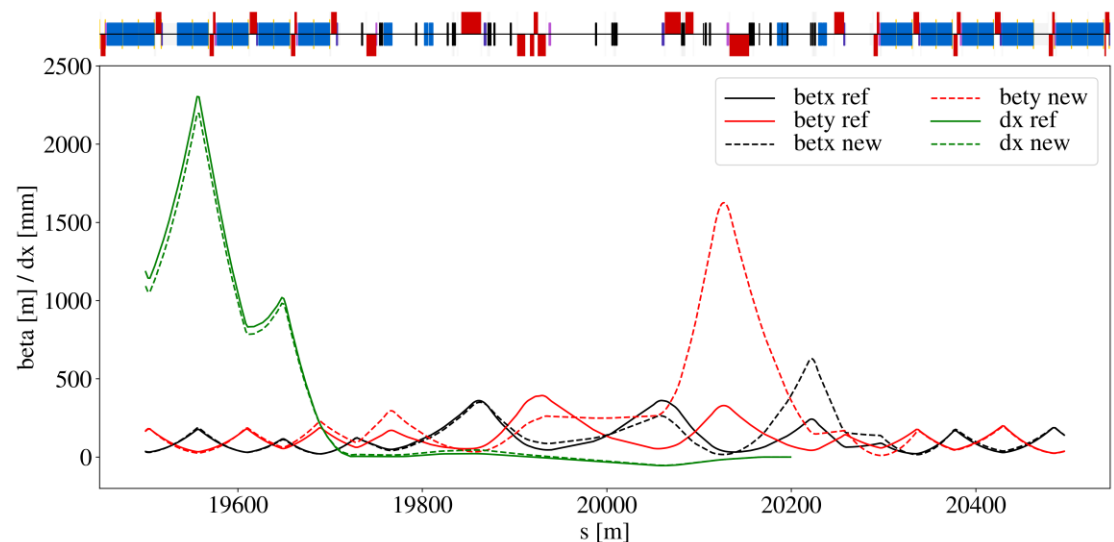
Xsuite provides a **multi-objective optimizer** to "match" model parameters to assigned constraints (e.g. control tunes, chromaticity, build orbit bumps, modify the optics)

- Based on the **extensive experience of MAD-X** → Uses the **same optimization algorithm** (Jacobian, proven robustness)
- The optimizer can be used to **synthesize knobs**
- Optimizations can involve **targets and knobs from multiple beam lines** (colliders)
- Interface **designed for usage flexibility**. User can **intervene in the optimization** by:
 - Enabling/disabling targets or knobs
 - Rolling back optimization steps
 - Changing knob limits, target values, convergence tolerances

Used for **optics matching of the LHC and of FCC-ee colliders**

→ Proved capability of handling **large problems** with several constraints and degrees of freedom.

Optimized optics for the LHC collimation area (designed with Xsuite)





- **Introduction**
 - Motivation
 - Design goals and constraints
 - Architecture
 - Agile development
- **Lattice modeling, simulation and optimization**
 - Lattice modeling
 - Single-particle tracking
 - Dynamic parameter control
 - Multi-objective optimizer
- **Simulation of specific processes**
 - Particle-matter interaction
 - Synchrotron radiation
 - Handling of collective effects
 - Space-charge
 - Beam-beam
 - Electron clouds
- **Final remarks**

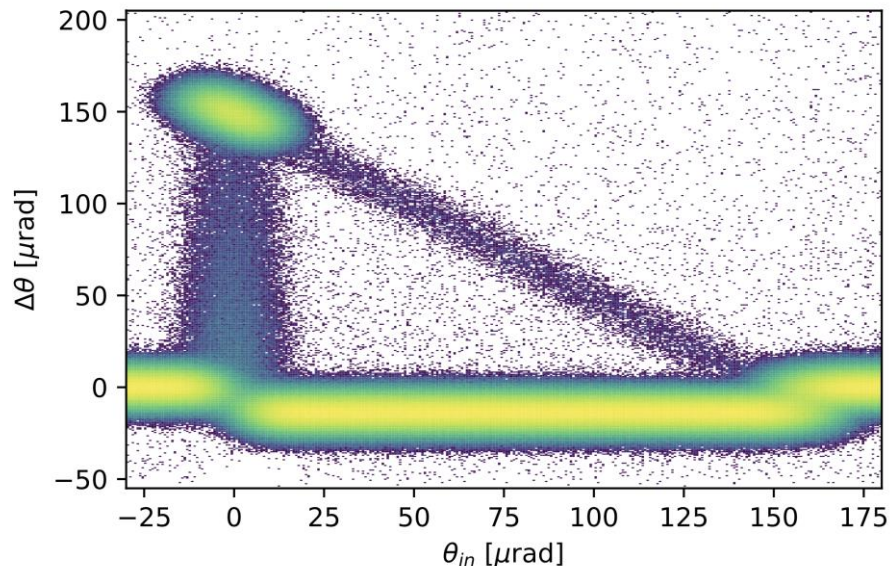
For collimation studies, the **Xcoll module** provides **three engines** to simulate **particle-matter interactions**:

- The **“Everest” engine** embedded in Xcoll (evolution of K2 module from Sixtrack)
- The **“Geant 4” engine**, based on an **interface Xsuite-BDSIM-Geant4**
- The **“FLUKA” engine**, based on an interface with the **FLUKA** Monte Carlo code

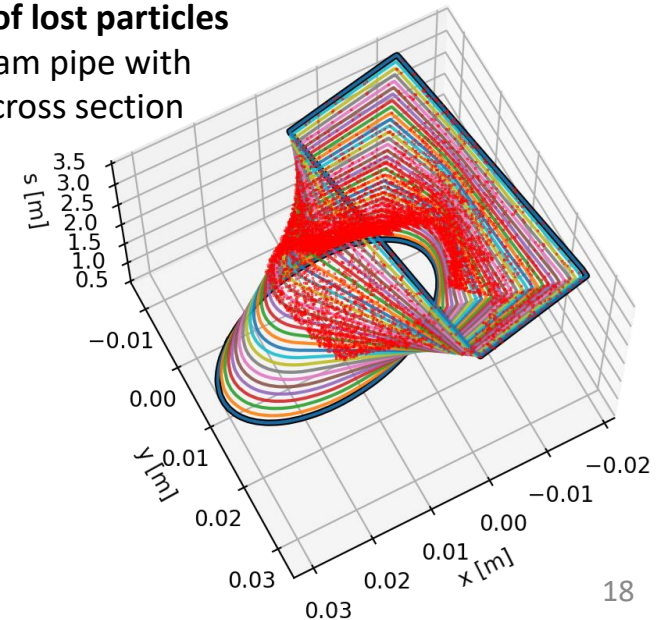
To support collimation studies, Xsuite provides:

- Tools to **automatically install and set collimators** in the simulation model
- Support for **complex aperture modelling** and **accurate localization of the lost particles along the beam line** (typically within 1-10 cm)

Particle deflection from a bent crystal (Everest engine)

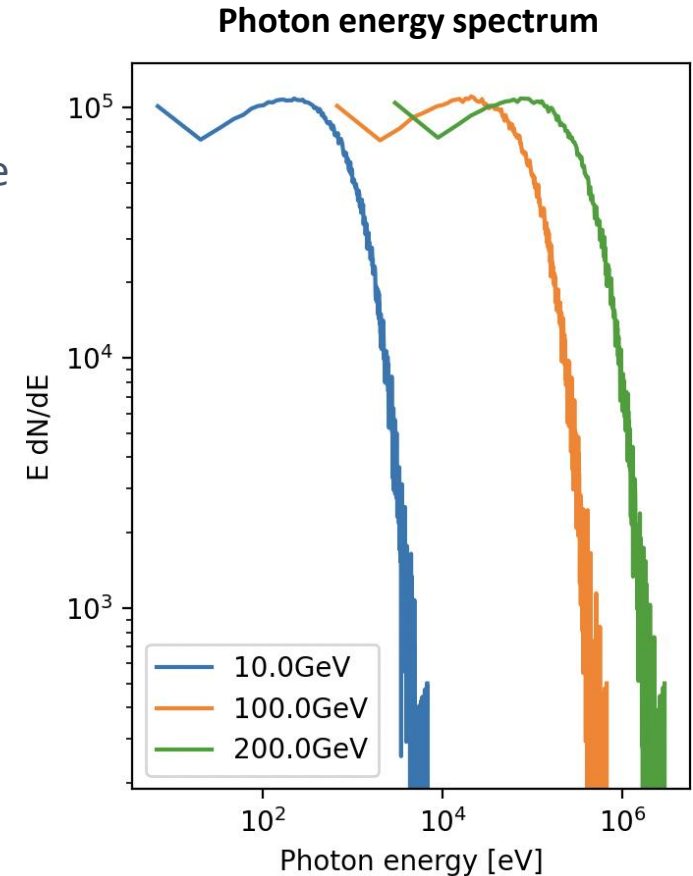


Localization of lost particles along a beam pipe with changing cross section



The effect of **synchrotron radiation** can be included in Xsuite tracking simulations. Two models available:

- The **“mean” model**, for which the energy loss from the radiation is applied particle by particle without accounting for quantum fluctuations;
- The **“quantum” model** for which the actual photon emission is simulated⁽¹⁾.



Plots courtesy Leon Van Riesen-Haupt, EPFL

⁽¹⁾ Based on H. Burkhardt, “Monte Carlo generator for synchrotron radiation”, 1990. Implementation ported from PLACET (A. Latina)

⁽²⁾ E. Forest, *From tracking code to analysis: generalised Courant-Snyder theory for any accelerator model*. Springer, 2016

The effect of **synchrotron radiation** can be included in Xsuite tracking simulations. Two models available:

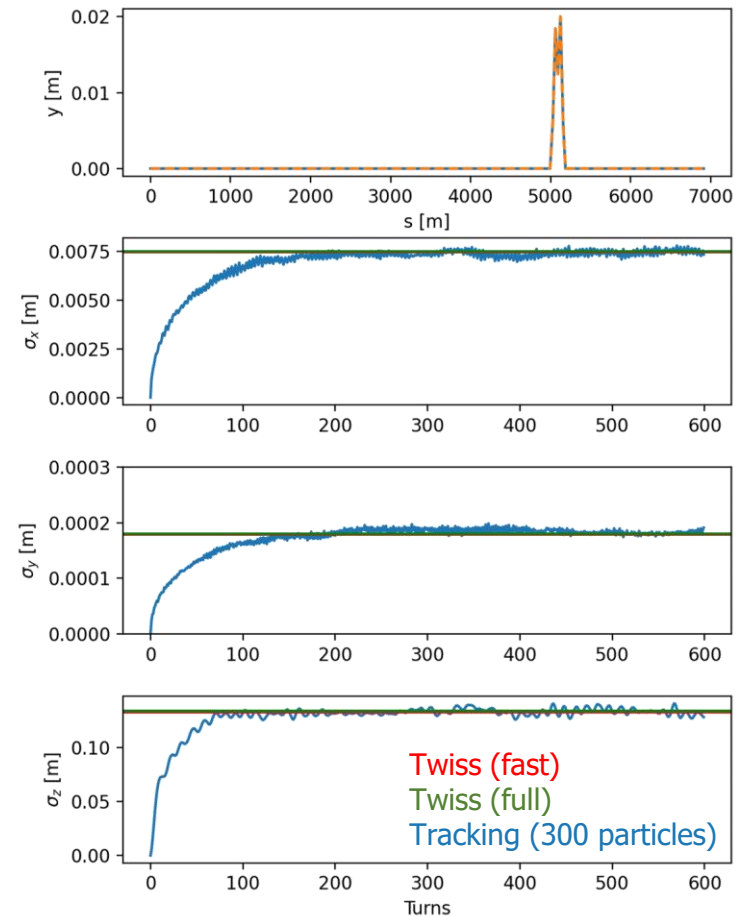
- The **“mean” model**, for which the energy loss from the radiation is applied particle by particle without accounting for quantum fluctuations;
- The **“quantum” model** for which the actual photon emission is simulated⁽¹⁾.

The **Xsuite Twiss** also includes:

- Dedicated algorithm for **non-symplectic one-turn map**⁽²⁾
- Computation of **radiation energy loss, damping times and equilibrium emittances**

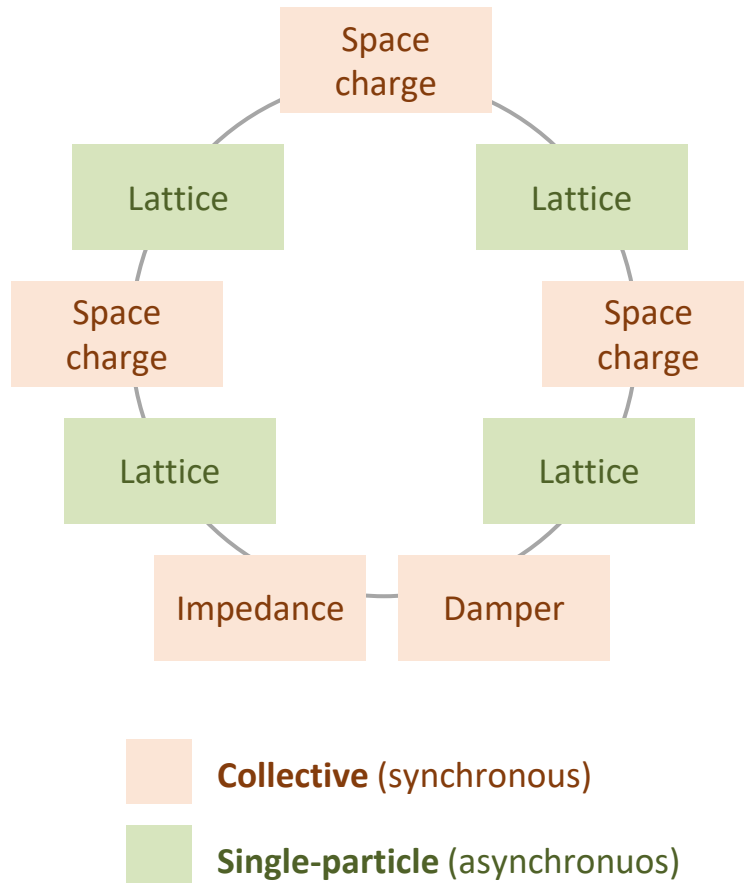
An **automatic tool** is provided for **phasing the RF cavities** and **adjusting magnet strengths** to **compensate the radiation energy loss (“tapering”)**

Equilibrium emittance (twiss vs track)



⁽¹⁾ Based on H. Burkhardt, “Monte Carlo generator for synchrotron radiation”, 1990. Implementation ported from PLACET (A. Latina)

⁽²⁾ E. Forest, *From tracking code to analysis: generalised Courant-Snyder theory for any accelerator model*. Springer, 2016



- Xsuite is designed to include **collective effects** in the simulations
- Handling of collective elements is **fully automatic** → The Xtrack module identifies the collective elements and **splits the sequence**:
 - The **non-collective** parts are handled **asynchronously** to **gain speed**
 - The simulation of the **collective effects** is **performed synchronously**
- **If requested** by the collective element, the particles' data is **automatically transferred from GPU to CPU and back**
- **Space-charge, beam-beam, e-cloud** (weak-strong) are handled **natively**
- **Impedances** and **feedback systems** are handled through an interface with **PyHEADTAIL**
- An automatic tool for the **computation of stability diagrams** from amplitude detuning is also provided

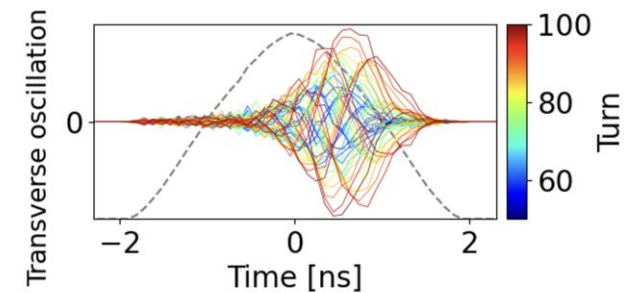
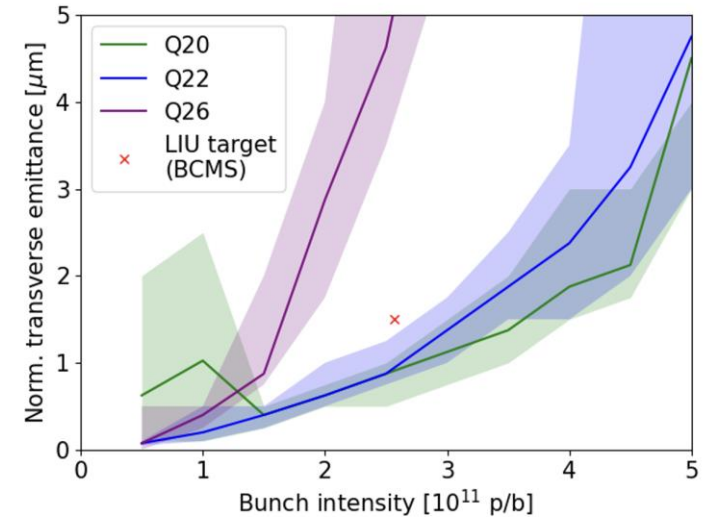
The implementation is largely based on

PyHEADTAIL-PyPIC

Different **space-charge models** are implemented:

- The **“frozen” model**, in which particles interact with fixed charge distributions
- The **“quasi frozen” model**, in which the **beam intensity and beam sizes are recomputed at each interaction**
- The **“Particle In Cell (PIC)” model**:
 - Charge of tracked particles distributed on a **rectangular grid**
 - **Fast Poisson solver** based on **FFT** method with Integrated Green Functions
- Space charge simulations **strongly profiting from GPU acceleration**

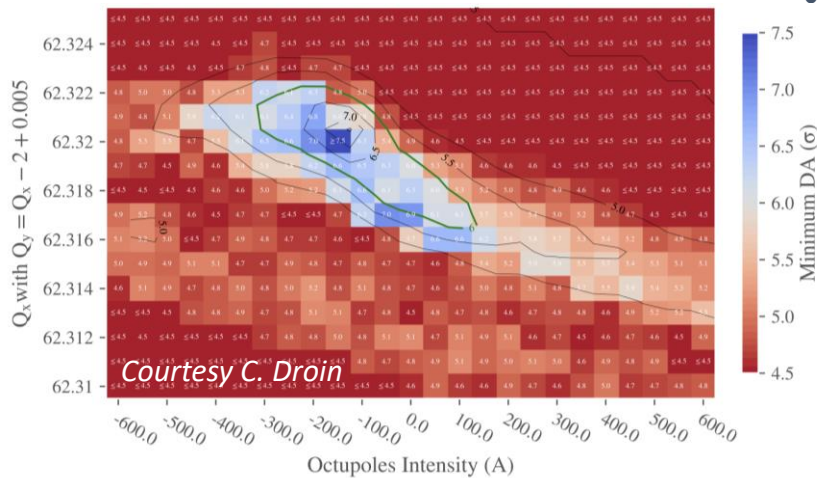
Simulation campaign for the CERN SPS including full non-linear lattice, space charge and wakefields



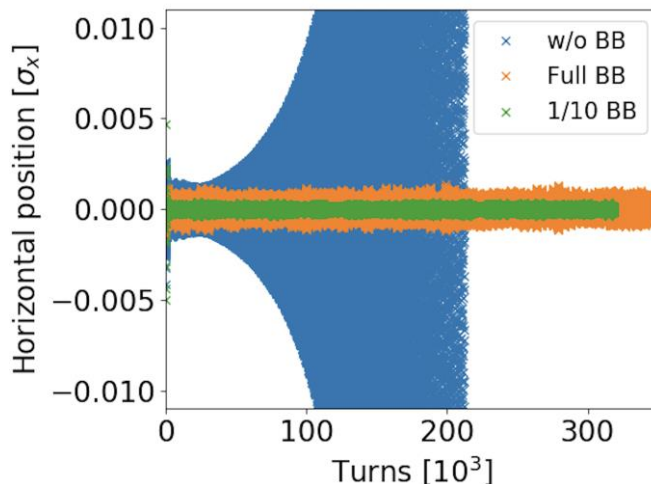
N. simulations	400
Number of PIC calculations per turn	540
Number of turns per simulation	40'000
Computing time per sim. (GPU)	~3 days
Computing time per sim. (CPU serial)	> 12 months

Courtesy X. Buffat

Dynamic aperture optimization study for HL-LHC (weak-strong modelling)



Wakefield + beam-beam simulations for HL-LHC (strong-strong modelling)



- Xsuite implementation based on experience from **Sixtrack** and **COMBI**
- Two models are provided:
 - The **“4D” model**, which applies **only transverse forces independent on the longitudinal motion**
 - The **“6D” model**, which applies **longitudinal and transverse forces** accounting for the synchrotron motion (method by Hirata et al.)

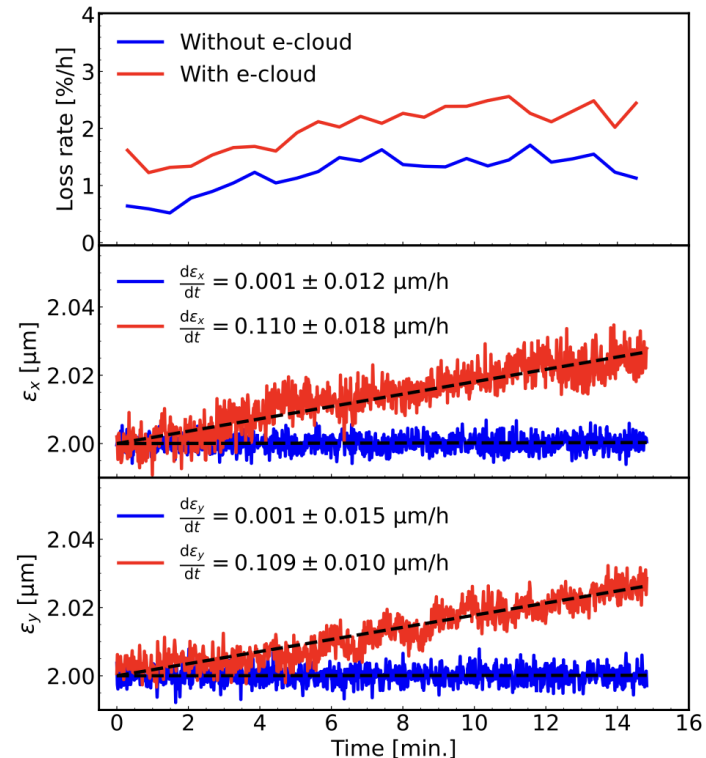
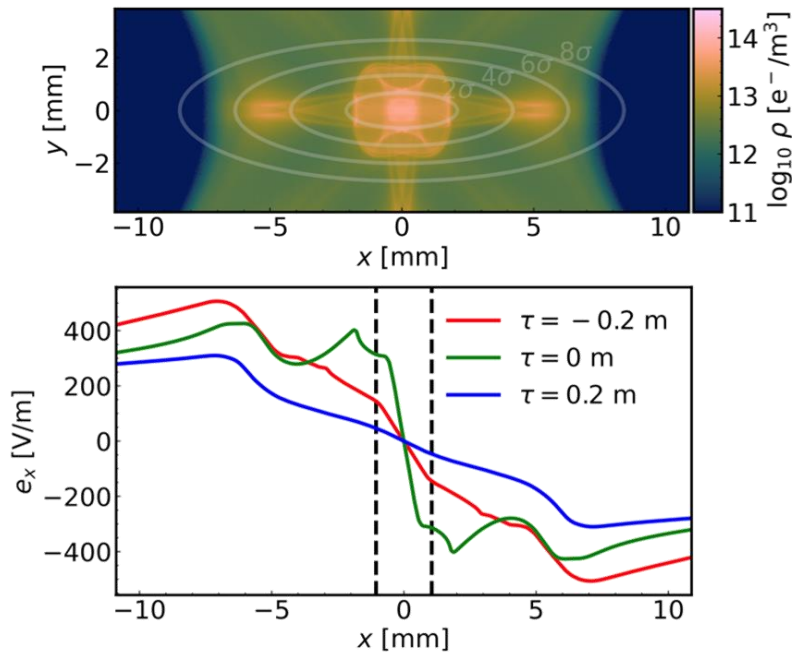
Both models can be used either in **“weak-strong” mode** (fixed assigned distribution for the other beam) or in **“strong-strong” mode** (self-consistent two-beam simulation)

- For the simulation of lepton colliders, the code can also simulate for **beamstrahlung** and **Bhabha scattering**
- **Strong-strong simulations** are accelerated by **parallel computing on HPC clusters** (based on MPI)
 - **“Pipeline” algorithm⁽¹⁾** used to optimize workload distribution across the nodes

⁽¹⁾ S. Furuseth and X. Buffat, *Comput. Phys. Commun.* 244 (2019)

Xsuite has been exploited to study the **effect of electron cloud on slow beam degradation** (emittance growth, lifetime degradation).

- Done by applying a **high-order interpolation scheme** to the **e-cloud potential imported** from a dedicated multipacting simulator.
 - Scheme designed to **preserve the symplecticity of the resulting map** by ensuring the global continuity of the potential and required derivatives.
- **Use of GPUs is mandatory** to simulate the required long time scales ($>10^6$ turns).





- **Introduction**
 - Motivation
 - Design goals and constraints
 - Architecture
 - Agile development
- **Lattice modeling, simulation and optimization**
 - Lattice modeling
 - Single-particle tracking
 - Dynamic parameter control
 - Multi-objective optimizer
- **Simulation of specific processes**
 - Particle-matter interaction
 - Synchrotron radiation
 - Handling of collective effects
 - Space-charge
 - Beam-beam
 - Electron clouds
- **Final remarks**



Who uses Xsuite and for which applications?

[Activities that are already in production, or for which testing work is ongoing]

LHC and HL-LHC

- **Tracking for DA** - lattice only (M. Le Garrec, E. Maclean, C. E. Montanari, T. Pugnati, D. Veres)
- **Crab-cavity noise studies**(A. Fornara, Uni. Manchester)
- **Beam-beam weak-strong** (G. Sterbini, S. Kostoglou, C. Droin, E. Lamb, D. Christie Uni. Manchester)
- **Beam-beam strong-strong** (X. Buffat)
- **Wire compensation studies** (P. Belanger, D. Kalchev TRIUMF)
- **Collimation studies** (F. Van Der Veken, N. Triantafillou, B. Lindstrom, M. D'Andrea, P. Hermes)
- **Optics/orbit matching, including RDT matching** (B. Lindstrom, K. Paraschou, C. Droin, S. Kostoglou, Y. Angelis)
- **Non-linear corrections calculation** (J. Dilly)
- **Hollow e-lens studies** (P. Hermes + students)
- **Incoherent e-cloud effects** (K. Paraschou)
- **Beam instrumentation studies** (D. Alves, K. Lasocha, SY-BI)

Injector complex (PSB, PS, SPS, LEAR, AD, ELENA)

- **Space-charge studies** (F. Asvesta, T. Prebibaj)
- **Cooling studies** (D. Gamba, P. Kruty)
- **Instabilities with wakes and space-charge** (X. Buffat)
- **Slow extraction studies at PS and SPS** (P. Arrutia Sota, T. Bass, F. Velotti, SY-ABT)
- **Ion studies** (E. Waagaard)
- **Beam transfer simulations** (F. Velotti, SY-ABT)
- **MTE simulations, optics+tracking** (A. Huschauer, starting)

Light sources (Petra IV, Elettra, Bessy III)

- **Tracking and DA**

FCC

- **FCC-ee optics matching, tapering + RF phasing** (M. Hofer, J. Keintzel, P. Kicsiny (L. Van Riesen-Haupt))
- **FCC-ee DA studies** (M. Hofer, L. Van Riesen-Haupt, EPFL)
- **FCC-ee and FCC-hh beam-beam-SS&WS** (P. Kicsiny, D. Di Croce, EPFL)
- **Beamstrahlung, Bhabha** (P. Kicsiny)
- **FCC-ee and FCC-hh collimation studies** (A. Abramov, G. Broggi)
- **FCC-ee vibration studies** (J.P. Salvesen, M. Hofer, LAPP Annecy collab.)
- **Compton scattering studies** (A. Abramov, I. Debrot, M. Hofer)
- **Collective effects** (M. Migliorati, A. Ghribi - Uni Sapienza, A. Rajabi – Desy)

GSI/FAIR

- **Space-charge simulations** (A. Oeftiger and team)
- **Slow extraction simulations - SIS18 and SIS100** (P. Niedermayer, C. Cortes)
- **BTF studies** (C. Cortes)

Medical accelerators

- **Slow-extraction for PIMMS** (R. Taylor)
- **Slow-extraction for MedAustron** (F. Kuehteubl, E. Renner, et al.)

BNL - RHIC and EIC

- **Optics calculations and tracking**
- **Collimation studies**

Fermilab – Main Injector, Recycler, IOTA

- **Tracking studies**
- **Impedance + space-charge simulations**

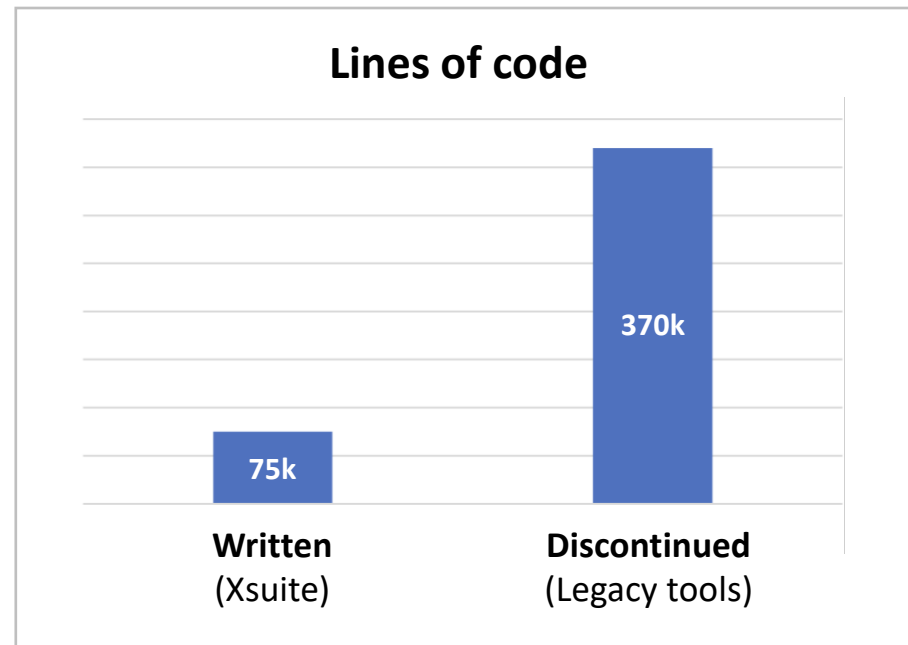
Training (University of Rome and EPFL)



In 2022-23 we have **essentially discontinued** the development and, to a very large extent, the usage of the following tools:

- Sixtrack
- Sixdesk
- Sixtracklib
- COMBI
- PySSD
- DistLib

This led to a **massive simplification** of our code base.





Some numbers:

Core team	3 persons
Contributing developers	16 persons
User community	~80 users
Publications including Xsuite simulations (in 2022-23):	25 papers

What next:

- **Most importantly:** user support, fixes and improvements based on user feedback, keep good quality documentation and testing
- **Additional features** (coming in 2024):
 - Equilibrium emittances in twiss in the presence of element shifts and tilts
 - Faster reload of deferred expressions (cython)
 - Native implementation of wakefields (with MPI parallelization)
 - RFTrack interface for tracking in fieldmaps
 - Intra-beam scattering model
 - Strong-strong beam-beam with PIC
 - Native save/load of sequence in ascii format (now done through MAD-X)



Thanks for your attention!