RECENT DEVELOPMENTS OF BLOND FOR NEW LARGE SCALE ACCELERATOR FACILITIES

S. Lauber*, S. Albright, C. Becker, H. Damerau, J. Flowerdew, N. Gallou, P. Hickersberger, L. Intelisano, B. E. Karlsen-Bæck, I. Karpov, E. Lamb, A. Lasheen, M. Marchi, O. Smedt, H. Timko, L. Valle, J. Wulff, M. Zampetakis,
European Organization for Nuclear Research, Geneva, Switzerland
M. Schwarz, Karlsruhe Institute of Technology, Karlsruhe, Germany
L. Thiele, University of Rostock, Rostock, Germany
R. Heine, Technische Universität Berlin, Berlin, Germany

Abstract

The Python-based Beam Longitudinal Dynamics (BLonD) simulation suite is an open-source framework for modelling the motion of charged particles in circular accelerators. BLonD has been in use since 2014 and is utilised successfully at several accelerator facilities such as J-PARC, ISIS, and CERN. Furthermore, BLonD is applied for the design of new synchrotrons such as the Future Circular Collider (FCC) and the Muon Collider. Both projects demand more complex simulations, as effects like strong synchrotron radiation or wake fields due to counter-rotating beams become important. In recent years, the BLonD community has also called for an improved user interface to ease the creation of input files for the simulations. This publication presents the latest software developments for BLonD. A redesigned, highly modular software architecture and improved user interface are currently being implemented, which simplify the creation of complicated relationships between physical phenomena. Preliminary testing and user feedback show improved scalability and flexibility, enabling the efficient development of particle-tracking simulations for next-generation accelerators, while facilitating the analysis of existing synchrotrons.

INTRODUCTION

With over a decade of development, the Beam Longitudinal Dynamics (BLonD) [1, 2] particle-tracking software is used to describe the evolution of a distribution of particles in the longitudinal phase space. BLonD was based on a modular structure to support the operation of existing accelerators, and to perform the longitudinal beam physics and radio frequency (RF) design of future accelerators. Over the years, it developed into a comprehensive software package that accounts for the interaction of the beam with the RF systems, RF noise, beam and cavity feedback, wake fields (in frequency and time domains), as well as synchrotron radiation and quantum excitation. Furthermore, it offers extensive support for generating initial distributions for single- and multi-bunch beams with arbitrary filling patterns and provides numerous implementations to model low-level radio frequency (LLRF) systems. Thanks to these features, BLonD has been employed to study longitudinal beam dynamics in numerous facilities [3–12].

BLonD is being upgraded from version 2 (V2) to version 3 (V3). The primary motivation for rolling out a new major release is the introduction of the so-called *Assembler* that links the various user-side classes into a consistent simulation setup. Consequently, there is an extensive number of Application Programming Interface (API) changes on the user side. To implement this functionality, monolithic classes had to be abandoned and replaced by a hierarchical, object-oriented system, which affects the majority of the code. This substantial reworking also allowed so-called type hints to be integrated into the major revision to simplify development. Furthermore, the ongoing research and development for the Future Circular Collider (FCC) [5,11,12] and the Muon Collider [6,13,14] required adjustments to the code architecture. This includes features such as counter-rotating beams, fast magnetic ramps, and sub-turn resolution of synchrotron radiation, which were only partially supported in BLonD V2.

CODE STRUCTURE

In its core routines, BLonD tracks the longitudinal macroparticle time offset Δt and energy offset ΔE on a turn-by-turn basis by modeling the user-selected interactions between the beam and the particle accelerator. First, the beam energy is updated according to the kick equation as a function of the longitudinal coordinate $\Delta E_{i+1} = \Delta E_i + f(\Delta t)$. The kick can contain energy changes due to the RF system, induced voltages, synchrotron radiation, etc. Second, the time coordinate is updated by the drift equation, based on the new energy coordinate as $\Delta t_{i+1} = \Delta t_i + g(\Delta E_{i+1})$. In addition, the properties of the accelerator, such as the RF frequency or the reference orbit, can be modified by the interaction with the beam.

The new structure of BLonD V3 can be analysed from various perspectives: the computational backends, the class-based middle layer, the abstract flow graph invoked by user input, and the public API classes. For the computational backends, the interaction between the beam's macroparticle coordinates and the various simulation components is central. For a single computing node Python (+ Numba [15] and Cupy [16]) and C++ (+ OpenMP [17] and CUDA [18])

* simon.fabian.lauber@cern.ch

kernels were implemented. Either, they transform the state of the beam coordinates ΔE and Δt, or convert them into properties such as the histogram of the beam current by time Δt, which is used to calculate collective effects such as the wake fields. Collective effects require additional treatment to ensure compatibility with distributed data processing. Since they inherently need to access all macroparticles across multiple compute nodes, the results from the individual nodes are aggregated via Message Passing Interface (MPI) into a consistent property such as a combined histogram across all compute nodes.

To enable code scalability, the computational backend is accessed through the new intermediate layer comprising abstract classes, reusable component classes for object composition, and inheritance. `AltersReference` provides the capability to change the reference coordinate system of the beam coordinates. `BeamPhysicsRelevant` adds a common interface for modifying ΔE and Δt. `Preparable` defines the interfaces for the initialization stage of the Assembler. `Schedulable` provides the generalised mechanism for changing parameters during the runtime, for example the RF phase of a cavity. Fig. 1 illustrates the user-relevant workflow.

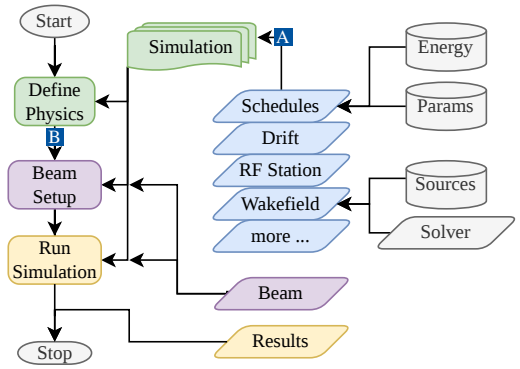

Figure 1: Abstract flow graph overview over the BLonD V3 user interaction with the code.

The backends and middle-layers are hidden, the public API classes are highlighted. To describe a beam physics case, the sequence of the various interactions with the beam must be defined (e.g. histogram → wake field → cavity kick → drift). Once the user has set up the simulation (see Fig. 1 A), a directed acyclic graph (DAG) is created from all nested components within `Simulation`, and implicit dependencies of attributes between the classes are automatically resolved. For example, the RF frequency depends on the ring circumference and the beam velocity. With the simulation as root node in the DAG, the different components are consistently accessible and transparent during the runtime. The initialised simulation defines the beam physics phenomena that the user intends to analyse. After the setup (see Fig. 1 B), several methods are available to generate an initial (matched or unmatched) macroparticle distribution. After selecting a beam distribution method that yields the coordinates Δt and ΔE, the simulation can finally be executed to provide insight into the dynamic behaviour of the system. BLonD V3 was extended to save the most common beam and component parameters to the hard drive and to prevent running into memory limitations towards the end of the runtime, which is critical for simulations that take hours to days.

Another decisive feature of the reworked software architecture is its improved extendability via predefined type-safe APIs. This makes it easy to write reliable extensions that integrate well within the BLonD runtime. Several classes have already been written by BLonD-users, for example a class to update the reference coordinate system at arbitrary times in the ramp, a class that combines kick, drift, and histogram calculation into a common class (which is used in the performance section of this publication), and advanced integrations to save results at runtime. The interaction of BLonD with Xsuite [19], the CERN 6D beam dynamics software, is currently being developed. This allows Xsuite to update Δt and BLonD to update ΔE, enabling to couple BLonD's comprehensive RF modelling features with Xsuite's transverse tracking.

QUALITY ASSURANCE

BLonD V2 has been thoroughly validated, also by extensive comparisons with theoretical predictions and measurement data from the aforementioned accelerators. The BLonD V3 test suite focuses on adding numerous unit tests, whilst the integration tests are designed towards ensuring internal coherence, as well as consistency with BLonD V2.

Software development requires repeated testing to ensure that physical correctness remains unchanged. There are a number of established industry standards, such as unit tests, integration tests, continuous integration and continuous deployment (CI/CD), version control (e.g. Git), code reviews, standardisation through automatic formatting, type safety, and documentation. All of them should be applied to ensure consistent, high-quality code and preserve maintainability.

For BLonD V3, these standards have been established primarily by blocking pre-commit hooks embedded in GitLab CI/CD. They include, amongst others, `ruff` and `numpydoc` for standardisation. The code-to-documentation ratio has been improved from 30 % to 70 % in BLonD V3. Around a thousand test cases are executed from Python 3.10 to 3.14 with all combinations of computing backends to ensure consistent results. They cover 100 % of the codebase. The test order is also randomised to detect unforeseen side effects. Altogether, the testing framework supports collaborative development for newcomers and experienced users alike. Currently, around 40 k test cases are run automatically per week using the GitLab CI/CD pipeline to enable asynchronous development of different parts of the code by multiple developers.

VALIDATION AND PERFORMANCE

The runtime of a simulation software is a central concern. It is one of the main factors determining how effectively users can conduct research and development to solve their real-world problems. The performance of BLonD V2 was already extensively improved and validated previously [20–

25]. However, BLonD V3 involves a comprehensive rewriting of many core parts and their interactions, leaving only the high-performance kernels untouched. Therefore, it is necessary to verify that physical accuracy and performance remain unaffected. For this purpose, one of the integration tests covering the main features of BLonD is presented. It includes solving the equations of motion for the particle drift along the synchrotron, acceleration through the RF cavities, the interaction of the beam with the effective beam-coupling impedance, as well as the scheduling of parameters, namely the energy ramp, the RF phase, and the momentum compaction factor.

As an example, the CERN Proton Synchrotron (PS) is used, as it has the most complex RF and beam manipulation system among the CERN synchrotrons. For illustration 7000 turns are tracked with a beam intensity of 2×10^{12} protons per bunch, represented by 2×10^{6} macroparticles. Only the impedance contribution of the main accelerating cavities (tuneable from 2.8 MHz to 10 MHz) is taken into account, with a beam histogram of 1024 bins and a cut-off frequency of 240 MHz for the cavity impedance. The complete input script is available via GitLab at Ref. [2, 26]. Figure 2 (top) shows the trajectory of a single particle in the

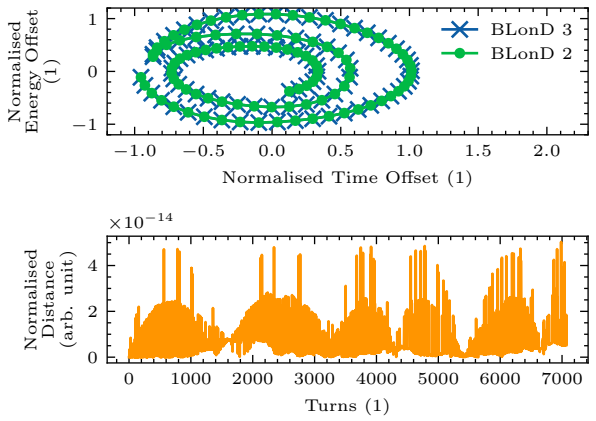

Figure 2: Comparison of BLonD V2 and BLonD V3 in the PS case, showing the evolution of the phase-space trajectory for a single macroparticle (top) and the normalised difference in coordinates as a function of turns between the versions (bottom).

longitudinal phase space tracked with both BLonD versions. To make the energy and time axes comparable, $\Delta t/30$ ns and $\Delta E/40$ MeV are plotted as normalised coordinates. From this, the normalised difference can be calculated to show the deviation between the two software versions. The residual is of the order of 10^{-14}, which is within the tolerance of a cumulative rounding error.

Further, the preservation of the BLonD performance is investigated. Fig. 3 illustrates the runtime differences across the various computing backends for the test case defined above; for other use cases, the performance bottlenecks could be computation- or memory-bound. The runtime of the Python backend is not shown, as it is not suitable for parallel execution. As expected, Numba is slightly slower than C++,

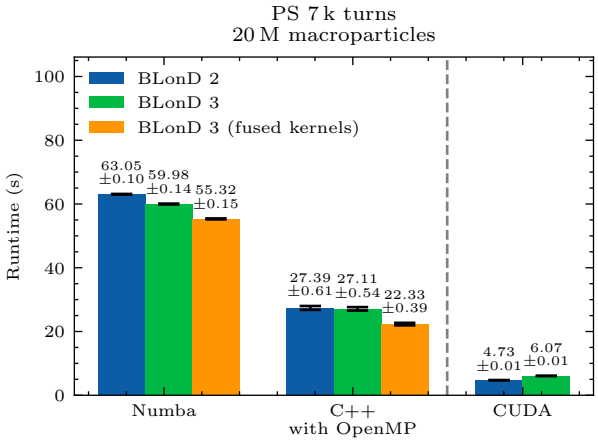

Figure 3: Preliminary performance comparisons of BLonD V2 and V3 with different backends (executed on Intel Xeon W-2125 or NVIDIA RTX A4000).

which in turn is outperformed by CUDA. The difference between the CPU- and GPU-based executions, as well as the absolute runtime, depends heavily on the hardware used. BLonD's performance remains virtually unchanged due to the already high degree of optimisation from the BLonD V2 backends which have been reused in BLonD V3. As the code architecture has been significantly modified, it is now possible to join simulation components and backend kernels into a single operation (e.g. kick, drift and histogram into a single kernel) without encountering compatibility issues when integrating these changes. By consolidating the most frequently executed operations, the speed of the computation for this standard use case can be further increased. The Numba kernel did not benefit from this improvement, as it used a different compiler and flags (LLVM instead of G++) and did not vectorise the instructions efficiently. The general runtime increase in BLonD V3 is an artefact of the architecture. Whilst BLonD V2 pre-computes all parameters as arrays for the entire acceleration ramp, BLonD V3 derives them just-in-time to remove the memory constraint for the number of turns. This shifts the computational cost from the initialisation phase to the turn-by-turn execution loop, resulting in slightly lower performance of the main turn-by-turn tracking loop.

CONCLUSION

The well-established longitudinal beam dynamics software suite for synchrotrons, BLonD, has reached its next major revision milestone V3. The so-called *Assembler* has been introduced to automatically set up the simulation and enhance user experience. Many code quality improvements have been made available, and the software architecture has undergone an overall rework. With the improved maintainability and extendibility, the collaborative development of new features and the integration with Xsuite is fostered. The accuracy and performance of the software remains uncompromised, and new options to further enhance the speed of the software were demonstrated in this publication.

REFERENCES

[1] H. Timko *et al.*, "Beam longitudinal dynamics simulation studies", *Phys. Rev. Accel. Beams*, vol. 26, no. 11, p. 114602, Nov. 2023. doi:10.1103/PhysRevAccelBeams.26.114602

[2] BLonD Collaboration, "BLonD — CERN code for the simulation of longitudinal beam dynamics in synchrotrons", GitLab repository, Accessed: 2026-03-26, 2026, https://gitlab.cern.ch/blond/BLonD,

[3] J. Coupard *et al.*, "LHC injectors upgrade, technical design report", CERN, Geneva, Switzerland, Rep. CERN-ACC-2014-0337, 2014. doi:10.17181/CERN.7NHR.6HGC

[4] O. S. Brüning *et al.*, "LHC design report", CERN, Geneva, Rep. CERN-2004-003-V-1, 2004. doi:10.5170/CERN-2004-003-V-1

[5] M. Benedikt *et al.*, "Future Circular Collider feasibility study report volume 2", *Eur. Phys. J. Spec. Top.*, vol. 234, no. 19, pp. 5713–6197, 2025. doi:10.17181/CERN.EBAY.7W4X

[6] C. Accettura *et al.*, "The Muon Collider", International Muon Collider Collaboration, Rep., 2025. doi:10.48550/arXiv.2504.21417

[7] H. Okita *et al.*, "Benchmarking of longitudinal calculation code BLonD for application to J-PARC RCS", in *Proc. 17th Annual Meeting of the Particle Accelerator Society of Japan (PASJ'20)*, Matsuyama, Japan, Sep. 2020, pp. 674–678. https://www.pasj.jp/web_publish/pasj2020/proceedings/PDF/FRPP/FRPP04.pdf

[8] H. Okita *et al.*, "Improvement of the longitudinal phase space tomography at the J-PARC synchrotrons", in *Proc. 14th Int. Particle Accelerator Conf. (IPAC'23)*, Venice, Italy, May 2023, pp. 4710–4713. doi:10.18429/JACoW-IPAC2023-THPL105

[9] M. Schwarz, "Haissinski solver for BLonD simulation suite", Research data, Dec. 2025, https://publikationen.bibliothek.kit.edu/1000189186,

[10] S. Albright, F. Antoniou, F. Asvesta, H. Bartosik, C. Bracco, and E. Renner, "New longitudinal beam production methods in the CERN Proton Synchrotron Booster", in *Proc. 12th Int. Particle Accelerator Conf. (IPAC'21)*, Campinas, SP, Brazil, May 2021, pp. 4130–4133. doi:10.18429/JACoW-IPAC2021-THPAB183

[11] L. Valle *et al.*, "RF power transients at injection energy in the FCC high-energy booster", unpublished.

[12] L. Valle *et al.*, "Energy ramps for the high-energy booster of the FCC-ee collider", unpublished.

[13] L. Thiele *et al.*, "Modelling of counter-rotating multi-turn wakefields in the Muon Collider RCS chain", unpublished.

[14] E. Lamb *et al.*, "Optimisation of transfer between RCS in the Muon Collider with synchronous phase", unpublished.

[15] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: a LLVM-based Python JIT compiler", in *Proc. 2nd Workshop on the LLVM Compiler Infrastructure in HPC (LLVM '15)*, New York, NY, USA, Nov. 2015. doi:10.1145/2833157.2833162

[16] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, "CuPy: a NumPy-compatible library for NVIDIA GPU calculations", in *Proc. Workshop on Machine Learning Systems (LearningSys) in the 31st Annual Conf. on Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, Dec. 2017. http://learningsys.org/nips17/assets/papers/paper_16.pdf

[17] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming", *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 1, pp. 46–55, 1998. doi:10.1109/99.660313

[18] NVIDIA, P. Vingelmann, and F. HP. Fitzek, "CUDA, release: 10.2.89", 2020, https://developer.nvidia.com/cuda-toolkit,

[19] G. Iadarola *et al.*, "Xsuite: an integrated beam physics simulation framework", in *Proc. HB'23*, Oct. 2023, pp. 73–80. doi:10.18429/JACoW-HB2023-TUA2I1

[20] H. Timko, J. Esteban Müller, A. Lasheen, and D. Quartullo, "Benchmarking the beam longitudinal dynamics code BLonD", in *Proc. IPAC'16*, May 2016, pp. 3094–3097. doi:10.18429/JACoW-IPAC2016-WEPOY045

[21] K. Iliakis, H. Timko, S. Xydis, and D. Soudris, "BLonD++: performance analysis and optimizations for enabling complex, accurate and fast beam dynamics studies", in *Proc. 18th Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS'18)*, Pythagorion, Greece, Jul. 2018, pp. 123–130. doi:10.1145/3229631.3229640

[22] K. Iliakis, H. Timko, S. Xydis, and D. Soudris, "Scale-out beam longitudinal dynamics simulations", in *Proc. 17th ACM Int. Conf. on Computing Frontiers (CF'20)*, New York, NY, USA, May 2020, pp. 29–38. doi:10.1145/3387902.3392616

[23] K. Iliakis, H. Timko, S. Xydis, P. Tsapatsaris, and D. Soudris, "Enabling large scale simulations for particle accelerators", *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4425–4439, 2022. doi:10.1109/TPDS.2022.3192707

[24] K. Iliakis, "Large-scale software optimization and micro-architectural specialization for accelerated high-performance computing", Ph.D. thesis, National Technical University of Athens, Greece, Feb. 2022. doi:10.12681/eadd/51282

[25] M. Migliorati and L. Palumbo, "Multibunch and multiparticle simulation code with an alternative approach to wakefield effects", *Phys. Rev. ST Accel. Beams*, vol. 18, no. 3, p. 031001, Mar. 2015. doi:10.1103/PhysRevSTAB.18.031001

[26] S. Lauber *et al.*, "BLonD v3 simulation input files for IPAC2026", GitLab repository, Accessed: 2026-03-26, 2026, https://gitlab.cern.ch/blond/blond3-simulations/publications/IPAC2026-Lauber,