The Journal of Open Source Software

Gyselalib++: A Portable C++ Library for Semi-Lagrangian Kinetic and Gyrokinetic Simulations

Emily Bourne[1], **Virginie Grandgirard**[2], **Yuuichi Asahi**[3], **Julien Bigot**[3], **Peter Donnel**[2], **Alexander Hoffmann**[4], **Abdelhadi Kara**[2], **Philipp Krah**[2], **Baptiste Legouix**[2], **Etienne Malaboeuf**[5], **Yann Munschy**[1,2], **Kevin Obrejan**[2], **Thomas Padioleau**[3], **Matthieu Protais**[2], **and Pauline Vidal**[4]

1 SCITAS, EPFL, CH-1015 Lausanne, Switzerland **2** CEA, IRFM, 13108 Saint-Paul-lez-Durance Cedex, France **3** Université Paris-Saclay, UVSQ, CNRS, CEA, Maison de la Simulation, 91191, Gif-sur-Yvette, France **4** Max-Planck-Institut für Plasmaphysik, Garching, Germany **5** CINES, 34000 Montpellier, France

DOI: 10.21105/joss.08582

Software

- Review
- Repository
- Archive

Editor: Pi-Yueh Chuang

Reviewers:

- @paulwilhelmvlasov
- @pratikvn

Submitted: 28 May 2025
Published: 09 September 2025

Summary

Gyselalib++ provides the mathematical building blocks to construct kinetic or gyrokinetic plasma simulation codes in C++, simulating a distribution function discretised in phase space on a fixed grid. It relies on the Discrete Domain Computation (DDC) library (Padioleau et al., 2025) to statically type the discretisation dimensions, thus preventing many common sources of errors. Via DDC, Gyselalib++ also leverages the Kokkos framework (Trott et al., 2022), ensuring performance portability across various CPU and GPU architectures. The library provides a variety of tools including semi-Lagrangian advection operators, quadrature rules, and solvers for elliptic and hyperbolic partial differential equations (PDEs). The majority of the operators are designed to work on non-orthonormal coordinate systems; those that don't use the static typing to raise compiler errors preventing their misuse.

Gyrokinetic Simulations

Plasma simulations are essential for the development of magnetic confinement fusion devices for energy production. The low collisionality of such plasmas makes kinetic models a judicious choice. In particular, gyrokinetic theory (Brizard & Hahm, 2007; Krommes, 2012), which reduces the 6D problem to a 5D problem by removing high-frequency gyromotion, is a popular framework for plasma simulation (Garbet et al., 2010). Despite the reduction in dimensionality, such simulations still require massively powerful high-performance computing (HPC) resources. For ITER[1]-sized simulations, exascale resources would still be required.

The pre-existing GYSELA code (Grandgirard et al., 2016), written in Fortran, originally aimed to simulate plasma in the core region of a tokamak using semi-Lagrangian advection with a distribution function discretised in phase space on a uniform grid. This approach was shown to work well, and it allowed the study of many interesting physical phenomena (Dif-Pradalier et al., 2022; Estève et al., 2018; Sarazin et al., 2021). However, expanding this code to use more complex mathematical methods such as non-uniform points (vital for handling the different magnitudes of physical quantities in the core and edge regions), and increasingly complex geometries (such as D-shape geometries, geometries including both open and closed field lines, X-points, and potentially stellarator geometries) has proved to be challenging and sometimes error-prone. These complexities are further amplified when trying to port such a code to GPU architectures, which are necessary for exascale simulations. This is a challenge shared by other gyrokinetic codes (Trilaksono et al., 2025).

[1] https://www.iter.org/

Statement of Need

In the case of GYSELA, the changes necessary to add non-uniform points and simplify the implementation of other new features, would have required an effort comparable to a complete rewrite, whereas actually performing such a rewrite brings additional benefits for design and portability. For example, we have been able to capitalise on C++'s strengths by using template programming to enforce the correctness of the implemented equations. A common source of error is writing equations with implicit assumptions, such as assuming an orthonormal coordinate system, or specific properties like those of a circular coordinate system. In Gyselalib++, equations are either expressed in tensor notation, so that they are either accurate for all geometries or do not compile, or they explicitly state their dependencies. C++ further enables us to add static assertions for cases with restricted applicability to prevent their misuse. Additionally, DDC is used to encode grid information directly in the type of each field, allowing the compiler to catch indexing errors at compile time. This is particularly useful when working with multiple grids along the same dimension, or when assigning different memory layouts to different fields.

In contrast to GYSELA, Gyselalib++ was conceived of and developed as a library, similar to the SeLaLib Fortran library (The SeLaLib project team, 2023), whose independent elements are each unit-tested and can be combined to build a final simulation. This design makes the library more versatile, enabling users to rapidly assemble a wide range of simulations, including high-dimensional test cases. The shared elements also provide more confidence in the reliability of the implementation, as they can prove their validity across multiple applications.

State of the Field

Most established gyrokinetic simulations, such as GENE-X (Michels et al., 2021) and GT5D (Idomura et al., 2009), are written in Fortran as stand-alone codes. This limits code sharing and reuse between projects. In contrast, many particle-in-cell codes, such as WarpX (Vay et al., 2018) and XGC (Ku et al., 2018), are now developed around reusable libraries like AMReX (Zhang et al., 2019) and Cabana (Slattery et al., 2022). To our knowledge, Gyselalib++ is the first such C++ library capable of Eulerian or semi-Lagrangian gyrokinetic applications. The Fortran library SeLaLib (The SeLaLib project team, 2023) plays a similar role in Fortran.

Contents

Gyselalib++ includes a range of reusable mathematical operators for plasma simulations. These include, but are not limited to, semi-Lagrangian advection schemes, numerical quadrature, differential operators (e.g. finite difference methods), solvers for common PDEs, and a multi-species collision operator (Donnel et al., 2019). A complete list of operators is available in the documentation[2]. Many of these tools are designed to work on a variety of grids, including non-uniform grids, which are especially important in edge-region simulations. The library also supports MPI-based parallelism, either with distributed operators or with transpositions between different multi-rank storage layouts.

The VOICE code (Bourne et al., 2023) has already been rewritten in C++ using the mathematical tools provided by Gyselalib++. Several common simulations including Landau damping (in 2D or in 4D MPI-parallelised Cartesian phase-space coordinates), a bump-on-tail instability (in 2D Cartesian phase-space coordinates), and a guiding-centre model (in polar coordinates) have also been implemented. While these examples are included primarily for illustration, they also serve as valuable testbeds for developing and validating new numerical methods. As such, these examples are ideal for mathematicians looking to validate new methods in realistic, publication-ready test cases.

[2] https://gyselax.github.io/gyselalibxx/

Acknowledgements

This project has received funding from the EoCoE-III project (Energy-oriented Centre of Excellence for Exascale HPC applications). This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Program (Grant Agreement No. 101052200 — EUROfusion). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them. Emily Bourne's salary was paid for by the EUROfusion Advanced Computing Hub (ACH). This project was provided with computer and storage resources by GENCI CINES thanks to the grant 2024-A0160502224 on the supercomputer Adastra's GENOA partition. We acknowledge the financial support of the Cross-Disciplinary Program on "Numerical simulation" of CEA, the French Alternative Energies and Atomic Energy Commission.

References

Bourne, E., Munschy, Y., Grandgirard, V., Mehrenberger, M., & Ghendrih, P. (2023). Non-uniform splines for semi-Lagrangian kinetic simulations of the plasma sheath. *Journal of Computational Physics*, *488*, 112229. https://doi.org/10.1016/j.jcp.2023.112229

Brizard, A. J., & Hahm, T. S. (2007). Foundations of nonlinear gyrokinetic theory. *Rev. Mod. Phys.*, *79*, 421–468. https://doi.org/10.1103/RevModPhys.79.421

Dif-Pradalier, G., Ghendrih, P., Sarazin, Y., Caschera, E., Clairet, F., Camenen, Y., Donnel, P., Garbet, X., Grandgirard, V., Munschy, Y., Vermare, L., & Widmer, F. (2022). Transport barrier onset and edge turbulence shortfall in fusion plasmas. *Communications Physics*, *5*(1), 229. https://doi.org/10.1038/s42005-022-01004-z

Donnel, P., Garbet, X., Sarazin, Y., Grandgirard, V., Asahi, Y., Bouzat, N., Caschera, E., Dif-Pradalier, G., Ehrlacher, C., Ghendrih, P., Gillot, C., Latu, G., & Passeron, C. (2019). A multi-species collisional operator for full-f global gyrokinetics codes: Numerical aspects and verification with the GYSELA code. *Computer Physics Communications*, *234*, 1–13. https://doi.org/10.1016/j.cpc.2018.08.008

Estève, D., Sarazin, Y., Garbet, X., Grandgirard, V., Breton, S., Donnel, P., Asahi, Y., Bourdelle, C., Dif-Pradalier, G., Ehrlacher, C., Emeriau, C., Ghendrih, Ph., Gillot, C., Latu, G., & Passeron, C. (2018). Self-consistent gyrokinetic modeling of neoclassical and turbulent impurity transport. *Nuclear Fusion*, *58*(3), 036013. https://doi.org/10.1088/1741-4326/aa6ebb

Garbet, X., Idomura, Y., Villard, L., & Watanabe, T. H. (2010). Gyrokinetic simulations of turbulent transport. *Nuclear Fusion*, *50*(4), 043002. https://doi.org/10.1088/0029-5515/50/4/043002

Grandgirard, V., Abiteboul, J., Bigot, J., Cartier-Michaud, T., Crouseilles, N., Dif-Pradalier, G., Ehrlacher, Ch., Esteve, D., Garbet, X., Ghendrih, Ph., Latu, G., Mehrenberger, M., Norscini, C., Passeron, Ch., Rozar, F., Sarazin, Y., Sonnendrücker, E., Strugarek, A., & Zarzoso, D. (2016). A 5D gyrokinetic full-f global semi-Lagrangian code for flux-driven ion turbulence simulations. *Computer Physics Communications*, *207*, 35–68. https://doi.org/10.1016/j.cpc.2016.05.007

Idomura, Y., Urano, H., Aiba, N., & Tokuda, S. (2009). Study of ion turbulent transport and profile formations using global gyrokinetic full-f Vlasov simulation. *Nuclear Fusion*, *49*(6), 065029. https://doi.org/10.1088/0029-5515/49/6/065029

Krommes, J. A. (2012). The gyrokinetic description of microturbulence in magnetized plasmas. *Annual Review of Fluid Mechanics*, *44*(Volume 44, 2012), 175–201. https://doi.org/10.1146/annurev-fluid-120710-101223

Ku, S.-H., Hager, R., Scheinberg, A., Dominski, J., Sharma, A., Churchill, M., Choi, J., Sturdevant, B., Mollén, A., Wilkie, G., Chang, C.-S., Yoon, E., Adams, M., Seo, J., Koh, S., D'Azevedo, E., Abbott, S., Worley, P. H., Ethier, S., … Chowdhury, J. (2018). *XGC*. [Computer Software] https://doi.org/10.11578/dc.20180627.11. https://doi.org/10.11578/dc.20180627.11

Michels, D., Stegmeir, A., Ulbl, P., Jarema, D., & Jenko, F. (2021). GENE-X: A full-f gyrokinetic turbulence code based on the flux-coordinate independent approach. *Computer Physics Communications*, *264*, 107986. https://doi.org/10.1016/j.cpc.2021.107986

Padioleau, T., Bigot, J., & Bourne, E. (2025). *DDC: The Discrete Domain Computation library*. https://doi.org/10.5281/zenodo.17078914

Sarazin, Y., Dif-Pradalier, G., Garbet, X., Ghendrih, P., Berger, A., Gillot, C., Grandgirard, V., Obrejan, K., Varennes, R., Vermare, L., & Cartier-Michaud, T. (2021). Key impact of phase dynamics and diamagnetic drive on Reynolds stress in magnetic fusion plasmas. *Plasma Physics and Controlled Fusion*, *63*(6), 064007. https://doi.org/10.1088/1361-6587/abf673

Slattery, S., Reeve, S. T., Junghans, C., Lebrun-Grandié, D., Bird, R., Chen, G., Fogerty, S., Qiu, Y., Schulz, S., Scheinberg, A., Isner, A., Chong, K., Moore, S., Germann, T., Belak, J., & Mniszewski, S. (2022). Cabana: A performance portable library for particle-based simulations. *Journal of Open Source Software*, *7*(72), 4115. https://doi.org/10.21105/joss.04115

The SeLaLib project team. (2023). *SeLaLib*. https://github.com/selalib/selalib

Trilaksono, J., Ulbl, P., Williams, J., Pfeiler, C.-M., Finkbeiner, M., Dannert, T., Laure, E., Markidis, S., & Jenko, F. (2025). OpenACC and OpenMP-Accelerated Fortran/C++ gyrokinetic fusion code GENE-X for heterogeneous architectures. *Proceedings of the Platform for Advanced Scientific Computing Conference*, 1–12. https://doi.org/10.1145/3732775.3733587

Trott, C. R., Lebrun-Grandié, D., Arndt, D., Ciesko, J., Dang, V., Ellingwood, N., Gayatri, R., Harvey, E., Hollman, D. S., Ibanez, D., Liber, N., Madsen, J., Miles, J., Poliakoff, D., Powell, A., Rajamanickam, S., Simberg, M., Sunderland, D., Turcksin, B., & Wilke, J. (2022). Kokkos 3: Programming model extensions for the exascale era. *IEEE Transactions on Parallel and Distributed Systems*, *33*(4), 805–817. https://doi.org/10.1109/TPDS.2021.3097283

Vay, J.-L., Almgren, A., Bell, J., Ge, L., Grote, D. P., Hogan, M., Kononenko, O., Lehe, R., Myers, A., Ng, C., Park, J., Ryne, R., Shapoval, O., Thévenet, M., & Zhang, W. (2018). Warp-X: A new exascale computing platform for beam–plasma simulations. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *909*, 476–479. https://doi.org/10.1016/j.nima.2018.01.035

Zhang, W., Almgren, A., Beckner, V., Bell, J., Blaschke, J., Chan, C., Day, M., Friesen, B., Gott, K., Graves, D., Katz, M., Myers, A., Nguyen, T., Nonaka, A., Rosso, M., Williams, S., & Zingale, M. (2019). AMReX: A framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, *4*(37), 1370. https://doi.org/10.21105/joss.01370