Parallel gyrokinetic simulations with Python

Emily Bourne[1], Yaman Güçlü[2]

[1] *Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany*

[2] *Max-Planck-Institut für Plasmaphysik, Boltzmannstr. 2, 85748 Garching, Germany*

In the field of scientific computing, two competing requirements constitute a challenge for efficient code development: on the one hand, algorithm prototyping needs to be fast, flexible and interactive; on the other hand, the target application often requires single-core optimization, shared-memory and MPI parallelization, and strict quality control. As a result, the so-called 'prototyping' and 'production' phases tend to be completely separate, using different programming languages and libraries, input/output facilities, unit tests and diagnostics. Moreover, once the production phase is initiated, it is very hard to make changes to the code architecture or the basic algorithms. For all these reasons, the transition from prototyping to production is a large burden, especially for small teams with limited resources.

In an attempt to smoothen such a transition, we have investigated the possibility of using Python for the whole development process, and of accelerating the most computationally intensive kernels with Fortran. The novelty of this approach is that the Python-to-Fortran translation is performed automatically by Pyccel, a Python 3 acceleration library [1] to which we provide static type information in the form of pragmas or function decorators. This allows us to maintain only one version of the code, which can be run as pure Python or in 'Fortran-accelerated' mode. The correctness of the Fortran modules is simply verified by the original suite of unit tests, and because the Fortran source code is available, further manual optimization is possible.

For a case study we have taken the semi-Lagrangian solution of the reduced ($\mu = 0$) gyrokinetic Vlasov equation in screw-pinch geometry, for which a pure Fortran simulation already exists within the Selalib library [2, 3], and we have developed PyGyro, a Python code that offers small but significant algorithmic improvements over the original simulation. After acceleration with Pyccel, an ion temperature gradient test-case with PyGyro runs approximately 20% slower than the original code on 1 node (32 processes), but can be distributed over a much larger number of nodes (128 instead of just 8). In this talk we will present PyGyro and its new parallelization algorithms, we will critically review the obtained performance, and we will discuss future improvements.

References

[1] PYCCEL. https://github.com/pyccel/pyccel

[2] SELALIB. http://selalib.gforge.inria.fr

[3] G. Latu, M. Mehrenberger, Y. Güçlü, M. Ottaviani, E. Sonnendrücker, J. Sci. Comput. **74** (2018) 1601-1650