FINAL REPORT PROGRAM LEFE

Program LEFE/ action MANU	Project Title NYLES, a new kernel for LES	Years 2019 – 2021
PI name, email and lab: Guillaume Roullet, roullet@univ-brest.fr, Laboratoire d'Océanographie Physique et Spatiale (UBO, CNRS, IRD, IFREMER), Brest Participating Laboratories : LOPS	Contribution to no other national or international program. Other funding sources : lab recurrent funding, access to the TGCC via the DARI project "HIRESTOPO" gen12051.	

Context The kernels of community numerical models result from numerical and design choices made decades ago, making them very difficult to test new ideas. NYLES aims a testing radically new ways of discretizing and coding, by writing a kernel from scratch.

Objectives / scientific questions In NYLES we explored two main questions : how to achieve 1) high accuracy with minimal dissipation, and 2) large speed, in terms of GFlops per second. A third aspect was to test whether a pure Python implementation could rival with Fortran.

Main results

1) High accuracy with minimal dissipation. We implemented an upwind 5th-order discretization on both the tracer (the density) and the momentum, in the 3D non-hydrostatic equations. The originality consists in doing the upwinding for the momentum, on the vortex-force term, that arises in the vector-invariant form, not on the momentum flux term. The associated implicit numerical dissipation provides just the right amount of kinetic energy dissipation for the simulation to remain smooth at all times. The discretization really acts as a subgrid-scale (sgs) closure on its own, allowing to get rid of any explicit sgs closure, such as the Smagorinsky or the Vreeman ones. Figure 1 illustrates how the model handles the 3D turbulence arising from the breaking of an interfacial wave. By testing the Green-Taylor experiment, we discovered that the model exhibited to much discrepancy compared to published results (too fast instability and not enough dissipation). Replacing the linear 5th-order discretization with a WENO 5-th order, fixed everything. Despite its computational cost, it turns out that having WENO schemes on both the tracer and the momentum is an amazing combo to integrate Navier-Stokes like fluid systems. This is the most significant achievement.
To make this point to the OA modeling community, we also coded a Rotating Shallow Water (RSW) kernel [Roullet & Gaillard 2020]. Because the 2D geometry is lighter than the 3D, and because there is no Poisson equation to solve in the RSW equations, we also implemented a mask system to allow for arbitrary shaped domains, and more importantly to illustrate how the lateral friction should be discretized in this vortex-force upwinding technique. Lateral friction is handled diagnostically by setting the vorticity value on the primal mesh vertices sitting on the boundary.

2) Large speed. Speed was key in the code design, from the first line to the last. Overall the code achieves around 2 GFlops s^{-1} per core, for one core and up to 1,000 cores (tested on irene at TGCC), while being in pure Python. The rescaled walltime per time iteration per grid point is of the order 500 ns (Figure 2). The speed is achieved with a series of choices rather than just one. First, all the metric terms are removed except where they should appear. This spares the number of multiplications and the amount of data exchanged with the memory. Second, all computational functions are compiled with the Numba module. Therefore, despite being written in pure Python, the code is compiled (with LLVM). A third technique was tested (and published) consisting in using duplicated or triplicated arrays to always work with arrays whose data are contiguous in memory.

3) Pure Python vs Fortran. At the time of the project end, we are now stepping back from the conclusions we reached in the paper. Even though 2 GFlops s^{-1} on one core is already a good performance, it is still small

compared to the theoretical peak performance (27 Gflops s^{-1}, for the cpu we used, whose clock ticks at 3.4 GHz). So is it possible to go faster than 2 GFlops s^{-1} ? The answer is yes! How? By making sure the code uses the SIMD (single instruction multiple data), usually triggered in Fortran with the "-march=native" compilation flag. These set of assembly instructions provides the 8 times speed up between the peak performance and the clock frequency. The problem is that Numba does not always use these instructions, or when it uses it, it is not as fast as Fortran. This will likely be fixed when Numba will be more mature. For now, a possibility is to write the computational functions in Fortran, to compile them in a shared library and to import them in Python. With this technique, I now achieve 12 GFlops s^{-1} on the same cpu. I now know, because I have the evidence, that a CFD code kernel, including a 3D Poisson solver, should deliver at roughly 40% of the peak performance. If in addition, all the clutter due to the metric terms is removed then a code should really go fast. In terms of absolute time, a reasonable target for a LES kernel is 4 10^{-7} s per time iteration, per grid cell, for a RK3 (three stages per time iteration), for a cpu whose peak performance is 27 GFlops s^{-1}.

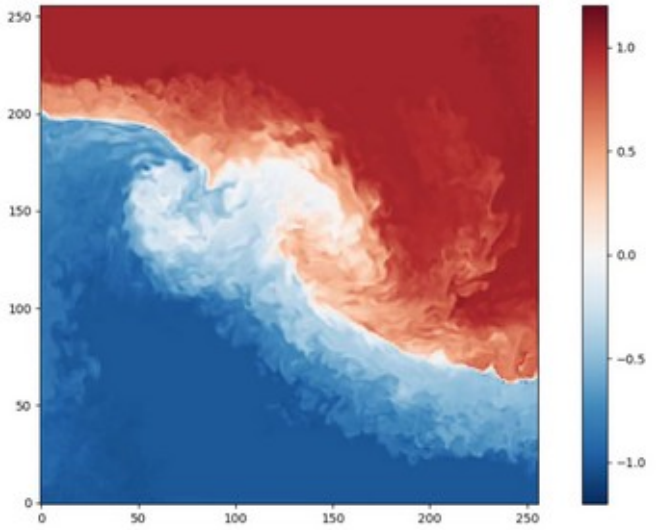

Figure 1.

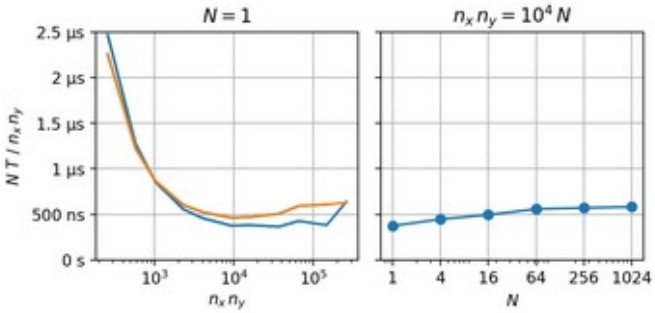

Figure 2

Figure 1 : Vertical section of buoyancy during the breaking of an interfacial wave (red lighter fluid, blue densier fluid, white mixed fluid resulting from diapycnal mixing). The domain was 256 x 64 x 256 (64 in the y-direction). The movie is available online.

Figure 2 : Wall time per time iteration T rescaled with the number of cores (N) and the number of grid points (nx ny). On the left for the mono-core case as a function of the domain size and on the right the weak scaling, where the number of cores N is increased from N = 1 up to N = 1,024, while the domain size per core nx ny = 10^4 is kept constant. In blue are the performances for the supercomputer and in orange for a notebook.

Future of the project :

It is now clear that community model kernels can/should be completely revised and that Fortran might be traded for a newer language such as Python or Julia. The idea is getting rapidly popular if one judges with these projects

1. ClimateMachine.jl, in Julia, driven by T. Schneider (Sridhar et al, 2021)
2. Oceananigans.jl, in Julia, supervised by MIT
3. ShallowWaters.jl, in Julia, supervised by M. Klöwer (Oxford) (Klöwer et al, 2021)
4. Veros, in pure Python but compiled with JAX, the Google compiler, a Python translation of the German model (Häfner et al, 2021)

The two lasts codes also share the same appetite for speed than NYLES, but exploring different ways. All these projects explore radical changes in their design, compared to the historical Fortran community models. However, it is worth emphasizing that the bigger the consortium (projects 1 and 2), the less readable, the more involved, the less simple, and likely the slower, is the code. I believe to be really competitive on tomorrow scene, a kernel should be developed by a very small team, of at most 4 people.

References :

- Häfner, D., Nuterman, R., & Jochum, M. (2021). Fast, cheap, and turbulent—Global ocean modeling with GPU acceleration in Python. Journal of Advances in Modeling Earth Systems, 13, e2021MS002717. https://doi.org/10.1029/2021MS002717
- Klöwer, M., Hatfield, S., Croci, M., Düben, P. D., & Palmer, T. N. (2022). Fluid simulations accelerated with 16 bits: Approaching 4x speedup on A64FX by squeezing ShallowWaters.jl into Float16. Journal of Advances in Modeling Earth Systems, 14, e2021MS002684. https://doi.org/10.1029/2021MS002684
- Sridhar, A., Tissaoui, Y., Marras, S., Shen, Z., Kawczynski, C., Byrne, S., Pamnany, K., Waruszewski, M., Gibson, T. H., Kozdon, J. E., Churavy, V., Wilcox, L. C., Giraldo, F. X., and Schneider, T.: Large-eddy simulations with ClimateMachine v0.2.0: a new open-source code for atmospheric simulations on GPUs and CPUs, Geosci. Model Dev. Discuss. [preprint], https://doi.org/10.5194/gmd-2021-335, in review, 2021.

Nombre de publications, de communications et de thèses

Roullet, G., & Gaillard, T. (2022). A fast monotone discretization of the rotating shallow water equations. Journal of Advances in Modeling Earth Systems, 14, e2021MS002663. https://doi.org/10.1029/2021MS002663.

An oral communication was scheduled at ETC 2021 (Zurich) but the conference has been cancelled because of the pandemic.

No PhD thesis