RESCIENCE C

Reproduction / Fluid dynamics

[¬Rp] Stokes drag on conglomerates of spheres

Konrad Hinsen[1,2]

[1]Centre de Biophysique Moléculaire, CNRS UPR4301, Orléans, France – [2]Synchrotron SOLEIL, Division Expériences, Gif sur Yvette, France

Edited by
Tiziano Zito

Reviewed by
Roman Yurchak

Received
04 March 2020

Published
11 June 2020

DOI
10.5281/zenodo.3889694

Introduction

The article that is the subject of this reproduction attempt [1] describes the validation (by comparison to experiments) of a new method for computing hydrodynamic interactions in colloidal suspensions [1] and its implementation as a published Fortran 77 library called HYDROLIB [2]. The latter is the first scientific software I ever published, at a time when this was a well-established possibility (the journal "Computer Physics Communications" was launched as early as 1969), but not yet common practice. The motivation for publishing this library was to make the method available to other researchers for inspection and use. Reproducibility was something that neither myself nor anyone else in the lab had ever heard of. The code specific to the validation paper was therefore not published. This turned out to be the major obstacle to reproducing the results of the original paper, as I was unable to find a copy of that project-specific code in my collection of old computers and backups.
The validation of the new method consisted of computing the sedimentation velocities of several assemblies of spheres for which experimental measurements were available [3]. These nine assemblies were regularly shaped conglomerates of two to 167 plastic spheres glued together. The description of the conglomerates provided by the experimenters was detailed enough to permit a computational reconstruction. The code written for the validation did exactly that: define the configurations of spheres for each of the nine conglomerates, then call a few subroutines from HYDROLIB to compute the sedimentation velocity.

Historical context

The research project on the computation of hydrodynamic interactions was conducted from 1992 to 1994 and combined theoretical work, software development, and computational evaluations such as those ultimately published in the article that is the subject of this reproduction attempt. The computers that were available to me at that time were a DECstation 5000/260 in the lab, an IBM mainframe computer at RWTH Aachen University's computing center, a Cray Y-MP at the nearby HLRZ Jülich (Höchstleistungsrechenzentrum, now called Jülich Supercomputing Centre), and an Atari TT at home. The only programming language available on all of these machines was Fortran 77, which was also the dominant language for scientific computing at the time. Fortran 77 was

Correspondence should be addressed to Konrad Hinsen (konrad.hinsen@cnrs.fr)
The authors have declared that no competing interests exists.
Code is available at https://github.com/khinsen/rescience-ten-year-challenge-paper-4. – SWH swh:1:dir:195b7890e84d3bcdc1fdbbbc351c63ac69a64ba0.
Open peer review is available at https://github.com/ReScience/submissions/issues/20.

defined by ANSI standard X3.9-1978, published in April 1978, to which specific compilers added various features that programmers had to avoid carefully in order to keep their code portable. My main development machine was the DECstation 5000/260, with regular tests performed on the three other computers to ensure portability.

A major limitation of standard-conforming Fortran 77 is the lack of both dynamic memory allocation and compile-time definitions. For HYDROLIB, this raised the question of how to deal with application-dependent parameters such as the number of particles, and with precision-related parameters (with significant impact on runtime and memory use) such as the multipole order to be used in approximations. Given the limited resources of smaller computers of the time (such as the Atari TT I used at home, with 32 MB of memory), conditional compilation was also desirable to prevent including unused code and data structures. The solution I adopted was not uncommon for the time: run all Fortran code through a C preprocessor before passing it to the Fortran compiler. Doing this portably and efficiently required scaffolding in the form of four scripts written in the `csh` language (which has since lost popularity but is still maintained) plus a `Makefile` for compiling the library code. The use of the C preprocessor for Fortran code was popular enough to be directly supported by some compilers at the time, and today's GNU Fortran still implements it. However, support was not universal. It was missing e.g. from the compilers provided by IBM and HP. My scripts therefore have an option for running the C preprocessor explicitly, producing a pure standard-conforming source code file specialized for an application's specific parameters that is then fed into the compiler.

The software development tools that I had at my disposal were Emacs and a source-level Fortran debugger. Version control already existed (CVS was published in 1990) but were not widely available and, more importantly, mostly unknown in computational physics. The same holds for automatized testing: the state of the art in the field was writing small example programs and running them manually from time to time. The most advanced aspect of my computing environment was backups: the lab's DECstation had a DDS-1 tape system, which made it straightforward to perform weekly backups of all user data.

Reproduction attempt

The GitHub repository for this reproduction attempt contains all the code, input data, and instructions for running the code on a modern GNU/Linux system with the Guix package manager [4, 5]. It also lists the exact version numbers of all the software involved in the reproduction. A total of 148 Guix package must be rebuilt identically to guarantee bit-for-bit reproduction of the results.

Finding the code

The main part of the code, HYDROLIB, is still available from the CPC Program Library [6] under the catalogue id ADBK, exactly as it was deposited in 1995. An updated version has been available on GitHub since 2015 [7]. The update does not modify the Fortran code, but only the compilation and installation scripts, which have in particular been adapted to the GNU Fortran compiler. The user guide has also been revised to make it compatible with modern LaTeX installations (a minor change to three lines of the preamble).

The search for the Fortran programs that compute the sedimentation velocities of the nine conglomerates was more difficult and ultimately failed. Of the four computers that I used at the time, only one has survived: my personal Atari TT. It is still in working condition, even though it tends to crash after about 30 minutes of operation, and has a copy of HYDROLIB on its disk, but not the programs for the nine conglomerates, most of which would have exceeded the computer's resources. The backup tapes from the DECstation have been lost during the intercontinental moves of my postdoc years.

The oldest backups that I could find in my collection are a DDS-1 tape from 1998 and a DDS-2 tape from 1999. Both are backups of the Linux PC that I used during my first years at the Centre de Biophysique Moléculaire in Orléans. There is a good chance that these tapes contain a backup of the programs, but I have not succeeded in finding a suitable tape reader. It is of course also questionable if these 20 year old magnetic tapes can still be read today. The lab had Linux PCs with DDS tape backup until about 2010, but they were progressively less and less used as the lab members moved to macOS. When the last of these PCs stopped working because of a disk failure, it was discarded without any further thought about the long forgotten collection of backup tapes.

Running the code

Because of the loss of all the conglomerate-specific programs, none of the results in the original article could be reproduced. This leaves the option of checking if HYDROLIB and its example application (a linear chain of spheres) can still be run on a modern computer. This is indeed the case and the file `notes.org` in the companion GitHub repository contains the detailed sequence of commands to compile and run the software reproducibly using the Guix package manager. It also contains the full list of dependencies, including the versions that were used.

Conclusion

The main utility of a failed reproduction attempt is the occasion it represents to learn from the mistakes of the past in order to do better in the future. From this perspective, I propose three conclusions:

Backups are not archives. Backups are safeguards against information loss due to device failures or operator errors, but only short-term. Backup technologies are designed for ease of use, not for longevity. In particular, backups suffer from the risks of material degradation (e.g. tapes not being readable any more) and technical obsolescence (devices or software for restoring the data may no longer be available). Archives, in contrast, are designed for ensuring the longevity of particularly important information, at the cost of a higher effort for storing and retrieving data. Archives are managed by professionals who know and continue to develop best practices for information preservation. Backups and archives are complementary. Back up everything, archive what is most important.

Publication no longer implies archiving. Archiving the scientific record has been a joint effort of scientific publishers and scientific libraries for centuries, and they have assumed this responsibility very well. Information technology has changed the scientific publication system profoundly and will continue to do so. Libraries have lost importance in the eyes of many scientists, and some would like to see the traditional publishers disappear completely because of their predatory attitude. However, it is important not to lose archiving in the process. Platforms or services for rapid sharing, such as preprint servers or collaborative coding platforms, generally don't provide archiving. On the other hand, new archiving platforms, such as Zenodo [8] or Software Heritage [9], offer interesting replacements. While the scientific publication landscape remains in flux, scientists have to be careful in ensuring that the information (articles, code, data) they disseminate in electronic form does get archived by a competent and reliable institution.

Standards ensure longevity. While this reproduction attempt ultimately failed because small but critical parts of the total source code were lost, the fact that the surviving archived core part of the code (HYDROLIB) still works illustrates the importance of standards for the longevity of electronic artifacts. HYDROLIB was written in strictly standard-compliant Fortran 77 for portability, in a world where computing hardware and systems software were more diverse than they are today. This portability has stood the test of time. In contrast, my other contribution to the ten-year-challenge [10] is about Python code that is half as old as HYDROLIB but no longer executable without modifications. The reasons for the shift from standardized to fragile technology as the basis for scientific software are too complex to discuss here, but it's clearly an aspect that the scientific community will have to consider more seriously in the future.

References

1. B. Cichocki, B. U. Felderhof, K. Hinsen, E. Wajnryb, and J. Bławzdziewicz. "Friction and Mobility of Many Spheres in Stokes Flow." In: **The Journal of Chemical Physics** 100.5 (Mar. 1994), pp. 3780–3790.
2. K. Hinsen. "HYDROLIB: A Library for the Evaluation of Hydrodynamic Interactions in Colloidal Suspensions." In: **Computer Physics Communications** 88.2-3 (Aug. 1995), pp. 327–340.
3. I. A. Lasso and P. D. Weidman. "Stokes Drag on Hollow Cylinders and Conglomerates." In: **Phys. Fluids** 29.12 (1986), p. 3921.
4. L. Courtès and R. Wurmus. "Reproducible and User-Controlled Software Environments in HPC with Guix." In: **European Conference on Parallel Processing**. Springer, 2015, pp. 579–591.
5. R. Wurmus, B. Uyar, B. Osberg, V. Franke, A. Gosdschan, K. Wreczycka, J. Ronen, and A. Akalin. "PiGx: Reproducible Genomics Analysis Pipelines with GNU Guix." In: **GigaScience** 7.12 (Dec. 2018).
6. **CPC International Program Library (1969-2016)**. http://cpc.cs.qub.ac.uk/. 2016.
7. [SW] K. Hinsen, **HYDROLIB**, 2019. LIC: BSD. SWHID: ⟨swh:1:dir:44d2058da36431234e4d6607f5ff001323472511;origin=https://github.com/khinsen/HYDROLIB⟩.
8. **Zenodo**. http://www.zenodo.org/.
9. J.-F. Abramatic, R. Di Cosmo, and S. Zacchiroli. "Building the Universal Archive of Source Code." In: **Commun. ACM** 61.10 (Sept. 2018), pp. 29–31.
10. K. Hinsen. "[Rp] Structural Flexibility in Proteins - Impact of the Crystal Environment." In: **ReScience C** (under review) (2020). swh:1:dir:8a73b8438b411cc8a58a3db7789ff45690808afa.