HAL
open science

FluidDyn: A Python Open-Source Framework for Research and Teaching in Fluid Dynamics by Simulations, Experiments and Data Processing

Pierre Augier, Ashwin Vishnu Mohanan, Cyrille Bonamy

▶ **To cite this version:**

Pierre Augier, Ashwin Vishnu Mohanan, Cyrille Bonamy. FluidDyn: A Python Open-Source Framework for Research and Teaching in Fluid Dynamics by Simulations, Experiments and Data Processing. Journal of Open Research Software, 2019, 7, 10.5334/jors.237 . hal-02088650

HAL Id: hal-02088650

https://hal.science/hal-02088650

Submitted on 10 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Journal of
open research software

Augier, P, et al. 2019 FluidDyn: A Python Open-Source Framework for Research and Teaching in Fluid Dynamics by Simulations, Experiments and Data Processing. *Journal of Open Research Software,* 7: 9. DOI: https://doi.org/10.5334/jors.237

SOFTWARE METAPAPER

FluidDyn: A Python Open-Source Framework for Research and Teaching in Fluid Dynamics by Simulations, Experiments and Data Processing

Pierre Augier[1], Ashwin Vishnu Mohanan[2] and Cyrille Bonamy[1]

[1] University of Grenoble Alpes, CNRS, Grenoble INP, LEGI, Grenoble, FR

[2] Linné Flow Centre, Department of Mechanics, KTH, Stockholm, SE

Correspondig author: Pierre Augier (pierre.augier@univ-grenoble-alpes.fr)

FluidDyn is a project to foster open-science and open-source in the fluid dynamics community. It is thought of as a research project to channel open-source dynamics, methods and tools to do science. We propose a set of Python packages forming a framework to study fluid dynamics with different methods, in particular laboratory experiments (package `fluidlab`), simulations (packages `fluidfft`, `fluidsim` and `fluidfoam`) and data processing (package `fluidimage`). In the present article, we give an overview of the specialized packages of the project and then focus on the base package called `fluiddyn`, which contains common code used in the specialized packages. Packages `fluidfft` and `fluidsim` are described with greater detail in two companion papers [4, 5]. With the project FluidDyn, we demonstrate that specialized scientific code can be written with methods and good practices of the open-source community. The Mercurial repositories are available in Bitbucket (https://bitbucket.org/fluiddyn/). All codes are documented using Sphinx and Read the Docs, and tested with continuous integration run on Bitbucket Pipelines and Travis. To improve the reuse potential, the codes are as modular as possible, leveraging the simple object-oriented programming model of Python. All codes are also written to be highly efficient, using C++, Cython and Pythran to speedup the performance of critical functions.

Keywords: Fluid dynamics research with Python; Numerical simulations; Laboratory experiments; Free and open-source software; modular; object-oriented; collaborative; efficient; tested; documented

Funding statement: This project has indirectly benefited from funding from the foundation Simone et Cino Del Duca de l'Institut de France, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 647018-WATU and Euhit consortium) and the Swedish Research Council (Vetenskapsrådet): 2013-5191. We have also been able to use supercomputers of CIMENT/GRICAD, CINES/GENCI (grant 2018-A0040107567) and the Swedish National Infrastructure for Computing (SNIC).

(1) Overview

Introduction

Science is mainly a collective activity. We can go further only by "standing on the shoulders of giants" (and of a huge number of technicians and scientists). Science is a lot about how to build new knowledge from the work of others and hence, the exchange of ideas is a fundamental aspect. In the last decades, we have lived through a revolution on how people exchange ideas. Computers of all kinds (from smartphones to HPC clusters) connected by a world wide web are used for human communication and also for many other applications. It has become nearly effortless to reproduce and exchange ideas and data. The set of intangibles that grows when shared and degrades when hoarded, such as knowledge and love, has been somehow extended. The information technology revolution open doors to fantastic opportunities for human collaboration, and of course, for science.

Web related activities alone account for at least 5% of GDP in the USA and the European Union.[1] A huge amount of money (and work) is invested on the related technologies. We are familiar with the most prominent companies involved in this dynamics (Google, Facebook, etc.), but there are also several smaller and sometimes lesser known organizations. Most of these companies base part of their work on the open-source paradigm contributing to open-source languages, libraries, software and operating systems, while also using them — a win-win situation for both the corporations and the community. This has lead to deep changes in software engineering, with a massive use of open-source methods and tools, for example distributed version control systems (DVCS) and web-based source development platforms.

The computer performance continues to increase exponentially, now also with the help of Graphical Processing Units (GPU). This gave way to a big boom in practical uses of data science and machine learning, which drives a strong research on artificial intelligence. Such developments contribute to progresses in open-source software. To summarize, there is a strong dynamics in play around the use of computers (in particular with the web) and this creates very efficient tools and methods for collective work and software development.

These changes in our world also reflect in the way science is done. Software and programming in science occupy a much bigger place than before. The role of software in science has changed. In the past, coding was sometimes considered as an inferior activity by some scientists. The focus was on the theory and the mathematical demonstration, which had to be elegant as it gets included in the articles. In contrast, it was normal to write crude code and to just show the results. Nowadays, codes tend to be at the heart of research.

"Open-science" is a new trend taking advantage of these new facts. Pioneering attempts are being made to do better science, improving reproducibility and collective efficiency, by using the open-source methods and tools for science and sharing and collaborating via the world wide web.

FluidDyn is a project to foster open-science and open-source coding in Python in the field of fluid mechanics. The project envisages to provide the technical framework to allow collaborative development of tools useful for the fluid mechanics community, and to do science with open methods. We provide examples of solutions for:

- Good coding practice with readable and easy to comprehend Python code (PEP 8).
- Source control management (Mercurial) and forge (Bitbucket) simple for the newcomers.
- Packaging and installation procedure. All packages are available from PyPI and can be install simply with pip, the standard Python installation tool.
- Licenses: depending on the packages, we choose to use the CeCILL-B or the CeCILL licenses. These licenses are compatible respectively with BSD and GPL licenses and adapted to both international and French legal matters.[2]
- Documentation produced with standard and up-to-date tools: Sphinx, Anaconda and Jupyter. Built and hosted online at Read the Docs. Can also be generated offline.
- Unittest and continuous integration with Bitbucket Pipelines and Travis.

We hope that such a clean framework will facilitate contributions from scientists in the field and that we can build together a nice, user-friendly and efficient ecosystem specialized in research and teaching in fluid dynamics.

Implementation and architecture

Organization of the code in packages

FluidDyn was originally intended to be a single package to perform experiments and simulations. Since a typical user may not be involved in both experiments and simulations and also, due to increasing complexity as a virtue of rapid development cycle made the need to decentralize FluidDyn evident. Now, FluidDyn project hosts a number of specialized packages, namely:

- `fluiddyn`[3]: The base package which contains pure-python code that can be reused in scripts or in specialized FluidDyn packages. It also contains codes for miscellaneous command-line utilities useful for a typical fluid dynamics user.

  The code of this package is presented in further detail in its documentation (https://fluiddyn.readthedocs.io/) and some prominent features are presented in the following subsection.
- `Transonic`: a pure Python package to accelerate Python-Numpy code with Pythran and potentially other Python compilers.
- `fluidfft` (see the companion paper [4]): a package which provides C++ and Python classes unifying various libraries to perform Fast Fourier Transform (FFT) in sequential and in parallel.
- `fluidsim` (see the companion paper [5]): Numerically oriented framework to run sequential and parallel Computational Fluid Dynamics (CFD) simulations and on-the-fly post-processing for a variety of problems (Navier-Stokes, Shallow Water, Föppl von Kármán equations, to name a few). A study using `fluidsim` has just been published in Physics of Fluids [3].
- `fluidlab`: Package to handle laboratory experiments. Primarily used to communicate with various hardware devices such as motors and pumps, to handle I/O between sensors, and to store data. Experiments using `fluidlab` have been carried out in the DAMTP fluid laboratory (Cambridge, UK. cf [2]), in ENS Lyon laboratory (Lyon, France [6]) and at LEGI (Grenoble, France [1]).
- `fluidimage`: Scalable image processing package which implements various algorithms to calibrate cameras, to preprocess images, to do Particle Image Velocimetry (PIV) and to postprocess data. `fluidimage` is used to process images taken during experiments performed in the Coriolis platform at LEGI [1].
- `fluidfoam`: Small package to load OpenFoam data and plot them.
- `fluidcoriolis`: Small package used to carry out experiments in the Coriolis platform (a large rotating platform participating in the European consortiums Euhit and Hydralab) and open the data obtained (see, for example [1]). One of the motivations behind creating this package is to study how to use open-source to create and share open-data.

A detailed presentation on the above packages can be found in their respective documentations on the web and for `fluidfft` and `fluidsim` in the two companion papers [4, 5]. The code base was designed to follow Python 2.7 syntax during its genesis. Now, it has been made forward compatible with Python 3 through the use of external package future. We now tend to abandon Python 2.7 support for the next releases of the package. This article will now focus on the base package `fluiddyn`.

API of the Python library `fluiddyn`

All functions and classes defined in `fluiddyn` are pure Python elements, meaning that no extensions are implemented in `fluiddyn`. Thus the package `fluiddyn` is extremely easy to install with just a `pip install` command and no compilation.

The package `fluiddyn` is organized into five sub-packages:

- `fluiddyn.io`: This subpackage provides utilities for input/output to different file formats.
- `fluiddyn.util`: Miscellaneous utilities.
- `fluiddyn.calcul`: Modules for numerical computations.
- `fluiddyn.clusters`: Classes to launch jobs on HPC clusters.
- `fluiddyn.output`: Utilities to produce scientific outputs (figures and videos).

Sub-packages `io`, `util` and `calcul` are the largest in terms of lines of code and provides the Application Programming Interfaces (API) to support `fluidsim`, `fluidlab` and `fluidimage`. For the sake of brevity, we shall only describe here some of the most important modules.

Module `fluiddyn.util.paramcontainer`

Sub-package `paramcontainer` defines class `ParamContainer` which is a hierarchical container for any type of parameters. As shown in this tutorial, various strengths of an object of this class include:

- Support containing and printing documentation on the parameters.
- Support printing to console as XML and saving as XML, HDF5 and NetCDF Files.
- Evaluate data types automatically while loading from saved files.
- Easy exploration in an interactive console.
- Allows modification of default parameters through simple Python script files. An error is raised if the user attempts to use an invalid parameter.
- Graphical User Interface (GUI) frontend with PyQt.

Thus, it makes it a much more robust implementation for saving key parameters, compared to conventional methods which rely on text or CSV files.

Module `fluiddyn.util.serieofarrays`

This module provides classes to iterate over files. It is a common task in data processing to understand how to organize a *sequence* of files from filenames and formats. Typically, one has to form smaller sets of arrays contained in the files. For example, we may have a *sequence* such as:

```
(im1_1.png, im1_2.png, im1_3.png, im2_1.png, im2_2.png, im2_3.png)
```

from which we could create different subsets like,

```
((im1_1.png, im1_2.png, im1_3.png),
(im2_1.png, im2_2.png, im2_3.png))
```

```
((im1_1.png, im2_1.png),
(im1_2.png, im2_2.png),
(im1_3.png, im2_3.png))
```

The classes of this module allows one to do it with a quite simple and general API as shown in the tutorial on the module.

Module `fluiddyn.util.mpi`

This simple module makes simultaneous sequential and MPI programming a breeze by providing number of processes, `nb_proc = 1` and `rank = 0`, when used in sequential mode, otherwise providing the appropriate values provided by `mpi4py` package. Use of this module thwarts coding several if-else clauses. If the program was using MPI, also defines the variable `comm` as an alias for the `MPI.COMM_WORLD` communicator.

Module `fluiddyn.calcul.easyfft`

Thin wrapper for an unified API using classes around the packages `pyfftw` and `scipy.fftpack`. It is very easy to perform forward and inverse Fast Fourier Transforms (FFT) in one-, two- and three-dimensions. The FFT can be multithreaded if the environment variable `OMP_NUM_THREADS` is defined.

The `fluiddyn` command-line utilities

The package `fluiddyn` also provides few command-line utilities to perform simple tasks useful for a scientist developing with Python and using the FluidDyn packages.

- `fluidinfo`
  Displays important information related to software and hardware. It includes detailed information such as currently installed FluidDyn packages, other third-party packages, C compiler, MPI and `Numpy` configuration.
- `fluiddump`
  Utility to print the hierarchy of HDF5 and NetCDF files. It does not depend on the NetCDF4 library.
- `fluidnbstripout`
  Very simple layer to stripout Jupyter notebooks of output, which is useful to keep the notebooks light-weight when included in a repository. This tool is based on nbstripout but, in contrast to nbstripout, by default the notebooks with a file name ending as '.nbconvert.ipynb' are excluded.
- `fluidcluster-help`
  Tiny utility to print a short documentation on the most useful commands to interact with the scheduler of a HPC cluster.
- `fluidmat2py`
  Utility to produce a strange code which is no longer Matlab and not yet Python. This strange code is then much easier to translate into correct Python than the original Matlab code.

Quality control

The package `fluiddyn` currently supplies unit tests covering around 70% of its code. These unit tests are run regularly through continuous integration on Travis CI with the most recent releases of `fluiddyn`'s dependencies and on Bitbucket Pipelines inside a static Docker container. The tests are run using standard Python interpreter with all supported versions.

For `fluiddyn`, the code coverage results are displayed at Codecov. Using third-party packages `coverage` and `tox`, it is straightforward to bootstrap the installation with dependencies, test with multiple Python versions and combine the code coverage report, ready for upload. It is also possible to run similar isolated tests using `tox` or coverage analysis using `coverage` in a local machine. Up-to-date build status and coverage status are displayed on the landing page of the Bitbucket repository. We also try to follow a consistent code style as recomended by PEP (Python enhancement proposals) 8 and 257. This is also inspected using lint checkers such as `flake8` and `pylint` among the developers. The code is regularly cleaned up using the Python code formatter `black`.

(2) Availability

Operating system

Windows and any POSIX based OS, such as GNU/Linux and macOS.

Programming language

Python 2.7, 3.4 or above. For the next versions, we will drop Python 2.7 support and Python >= 3.6 will be required.

Dependencies

We list here only the dependencies of the base package `fluiddyn`.

- **Minimum:** `Numpy`, `Matplotlib`, `psutil`, `future`, `subprocess32` (for Python 2.7 only), `h5py`, `h5netcdf`.
- **Full functionality:** `mpi4py`, `Scipy`, `pyfftw` (requires FFTW library), `pillow`.
- **Optional:** OpenCV with Python bindings, `scikit-image`.

List of contributors

- Pierre Augier (LEGI): creator of the FluidDyn project, developer of majority of the FluidDyn packages, future-proofing with Python 3 compatibility and documentation.
- Ashwin Vishnu Mohanan (KTH): developer of the packages `fluidsim`, `fluidfft`, `fluidimage` and `fluiddyn`; documentation, code coverage and continuous integration (Docker, Bitbucket Pipelines and Travis CI).
- Cyrille Bonamy (LEGI): developer of `fluidfft`, `fluidsim`, `fluidimage` and `fluidfoam`. Main maintainer of `fluidfoam`.
- Antoine Campagne (LEGI): developer of `fluidimage` and `fluidlab`.
- Miguel Calpe Linares (LEGI): developer of `fluidsim` and `fluidlab`.
- Julien Salort (Laboratoire de physique, ENS de Lyon): developer of `fluidlab`.

Software location

Name: PyPI
Persistent identifier: https://pypi.org/project/fluiddyn
Licence: CeCILL-B, a BSD compatible French licence.
Version published: 0.2.4
Date published: 02/07/2018

Code repository

Name: Bitbucket
Persistent identifier: https://bitbucket.org/fluiddyn/fluiddyn
Licence: CeCILL-B
Date published: 2015

Language

English

(3) Reuse potential

As a library, `fluiddyn` has been used in the project's specialized packages. The common code base for packages with such varied applications is a proof of `fluiddyn`'s versatility and generality. It can be used in other packages outside the FluidDyn project easily depending on the need. The command-line tools can be useful to all scientists working with Python. Other use cases could be:

- `ParamContainer` as a generic parameter storage standard.
- Modules within `fluiddyn.util` for miniature tasks, printing in terminal with colours, getting memory usage information, detecting if the current Python session is inside IPython, creating a string with time and date, etc.
- Modules within `fluiddyn.io` to read and save images of various formats including `TIFF` files with multiple images; easily ask user with yes/no queries; handle different file formats using classes, including CSV, HDF5, Digiflow, Dantec formats.
- Modules inside `fluiddyn.clusters` subpackage to make job submission scriptable in HPC clusters with OAR or SLURM job schedulers. It could be even used for non-Python jobs.

There is no formal support mechanism. However, bug reports can be submitted at the Issues page on Bitbucket. Discussions and questions can be aired on instant messaging channels in Riot (or equivalent with Matrix protocol)[4] or via IRC protocol on Freenode at `#fluiddyn-users`. Discussions can also be exchanged via the official mailing list.[5]

Conclusions

FluidDyn is an attempt to set off collaborative dynamics based on open-source development in fluid dynamics research. We shall try, with this project, to explore the possibilities of open-source in science and fluid dynamics by fully exploiting the new open-source tools and methods.

The project is right now in a preliminary stage. Packages are actively evolving with interesting features and a framework for collaborative development, packaging, documenting and testing is now well set. However, the community around the project is currently tiny and now, we have to work on attracting users and developers since an active community is a criteria for success and sustenance of an open-source project.

Will people use the FluidDyn tools and collaborate through the project FluidDyn? There are clearly many challenges and potential barriers:

- Some habits in the community.
- Knowledge and skills in the community. For example, only few people use issue tracker and pull requests.
- Lack of a business model for open-source software in science. A good quality open-source software has a cost that institutions should be willing to fund and support.
- Lack of recognition of the work spent in open-source, in particular for scientists without permanent positions.

On our side, we also have positive points. The quality of the tools we use (Python and its scientific ecosystem, Mercurial, Read the Docs, Jupyter, …) is impressive. Scientific code is done to be read and to transmit ideas. To this effect, Python is among the best languages today. Python starts to be a standard tool in fluid dynamics, especially used for CFD (Dedalus, SpectralDNS, TriFlow, PyLBM, Oasis, PyFR, FEniCS, Cassiopee, pyCGNS, etc.) and data analysis (OpenPTV, PyPIV). Moreover, we can benefit from the dynamics of Python and of emerging subjects like deep learning and the Internet of Things. Finally, if we manage to gather a community of users and of developers, the collective efficiency related to open-source methods and tools can be a strong booster.

Notes

[1] See the report by Internet Association titled "Refreshing Our Understanding of the Internet Economy".

[2] http://www.cecill.info/licences.en.html.

[3] We use FluidDyn (with capital letters) to name the project and `fluiddyn` for the base package.

[4] https://matrix.to/#/#fluiddyn-users:matrix.org.

[5] https://www.freelists.org/list/fluiddyn.

Acknowledgements

We thank the CNRS to finance the work of Pierre Augier while giving freedom in terms of scientific project. Similarly, Ashwin Vishnu Mohanan could not have been as involved in this project without the kindness of Erik Lindborg. We thank Antoine Campagne, Miguel Calpe Linares and Julien Salort for their implications in the development of some of the FluidDyn packages. We are grateful to Bitbucket for providing us with a high quality forge compatible with Mercurial, free of cost. Finally, we thank Gabriel Moreau and Joël Sommeria for their constant will to discuss and share their knowledge.

Competing Interests

The authors have no competing interests to declare.

References

1. **Campagne, A, Alfredsson, H, Chassagne, R, Micard, D, Mordant, N, Segalini, A, Sommeria, J, Viboud, S, Mohanan, A V, Lindborg, E** and **Augier, P** 2016 First report of the MILESTONE experiment: Strongly stratified turbulence and mixing efficiency in the coriolis platform, VIIIth International Symposium on Stratified Flows (ISSF, San Diego, USA).
2. **Leclercq, C, Partridge, J L, Augier, P, Dalziel, S B** and **Kerswell, R R** 2016 'Using stratification to mitigate end effects in quasi-Keplerian TaylorCouette flow'. *J. Fluid Mech*, 791: 608–630. DOI: https://doi.org/10.1017/jfm.2016.44
3. **Lindborg, E** and **Mohanan, A V** 2017 'A two-dimensional toy model for geophysical turbulence'. *Phys. Fluids*, 29(11): 111114.
4. **Mohanan, A V, Bonamy, C** and **Augier, P** 2019a 'FluidFFT: Common API (C++ and Python) for Fast Fourier Transform libraries'. *J. Open Research Software* (DOI).
5. **Mohanan, A V, Bonamy, C** and **Augier, P** 2019b 'FluidSim: Modular, objectoriented python package for high-performance CFD simulations'. *J. Open Research Software* (DOI).
6. **Salort, J, Rusaouëen, É, Robert, L, Du Puits, R, Loesch, A, Pirotte, O, Roche, P-E, Castaing, B** and **Chillà, F** 2018 'A local sensor for joint temperature and velocity measurements in turbulent flows'. *Review of Scientific Instruments*, 89(1): 015005. DOI: https://doi.org/10.1063/1.4989430

How to cite this article: Augier, P, Mohanan, A V and Bonamy, C 2019 FluidDyn: A Python Open-Source Framework for Research and Teaching in Fluid Dynamics by Simulations, Experiments and Data Processing. *Journal of Open Research Software,* 7: 9. DOI: https://doi.org/10.5334/jors.237

Submitted: 02 July 2018 **Accepted:** 15 February 2019 **Published:** 01 April 2019

Copyright: © 2019 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See http://creativecommons.org/licenses/by/4.0/.

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS