Capytaine: a Python-based linear potential flow solver

Matthieu Ancellin[1] **and Frédéric Dias**[1]

1 UCD School of Mathematics and Statistics, University College Dublin, MaREI Centre, Ireland

DOI: 10.21105/joss.01341

Software

- Review
- Repository
- Archive

Submitted: 19 March 2019
Published: 10 April 2019

Context

One of the first phases of the design of any ship or floating structure is its simulation with a linear potential flow model. This model gives a description of the interaction between the ocean waves and the floating body (Newman, 1977).

Assuming an incompressible, inviscid and irrotational flow, the fluid motion can be modeled by a scalar field ϕ, called velocity potential and such that $u = \nabla\phi$. The conservation of the mass of the fluid can be written as a Laplace problem $\nabla^2\phi = 0$. Assuming small motions of the floating body and small wave heights, the boundary condition at the free surface can be linearized. The resulting problem is fully linear and can be solved in the frequency domain.

The problem can be formulated as a boundary integral problem and solved by the Boundary Element Method (BEM). In contrast with other BEM problems, the linear potential flow solvers use an expression of the Green's function which accounts for the boundary conditions at the free surface and the sea bottom. Thus, only the wet surface of the floating body needs to be discretized. The Green's function can be computed in several ways (Xie et al., 2018), which are implemented in several software, most of them commercial and closed-source.

In 2014, the École Centrale de Nantes (ECN) released the code *Nemoh* as an open-source software (Babarit & Delhommeau, 2015). This solver is based on in-house and commercial codes developed in the ECN since the 1970's (Delhommeau, 1987, 1989, 1993). Since its open-source release, it has been used in numerous applications, in particular by the community of marine renewable energies (Penalba Retes, Kelly, & Ringwood, 2017).

However, the capabilities of the code are limited in comparison with its commercial counterparts. Developing new features is difficult since the core of the implementation dating back from the 1970's is poorly readable and the documentation is scarce. The goal of the present work is the modernization of this code in order to have an open-source linear potential flow solver that is easier to maintain and develop.

The `capytaine` Python package

The present work is a complete modernization of the Nemoh code, including a refactoring of the core routines and a Python user interface.

The core routines have been kept in Fortran, although a more modern coding style has been used. Dead code has been removed, no global variables are used anymore, more meaningful names have been used for functions and variables and a lot of comments have been added.

This new version is meant to be used within the scientific Python ecosystem. The integration of the Fortran core with Python has been done with F2PY (Peterson, 2009). The naive linear solver included in Nemoh has been replaced by a call to the state-of-the-art solvers from the Numpy and Scipy libraries (Jones, Oliphant, Peterson, & others, 2001; T. Oliphant, 2006). The Intel MKL library can thus be used to solve the linear system in parallel with OpenMP. The rest of the code is the independent computation of the coefficients of a matrix and has been straightforwardly parallelized with OpenMP, making most of the code parallel.

The code has also been integrated with other Python libraries. Reading, writing and transforming meshes is done by the integration of the `meshmagick` (Rongère, 2017) library. The default output format is a `xarray.Dataset` (Hoyer & Hamman, 2017) that can be saved in the standard NetCDF format. `VTK` (Schroeder, Martin, & Lorensen, 2006) is used for the 3D visualization of the meshes and the results. Testing of the code is done with the `pytest` (Krekel et al., 2004) library.

Efforts have been made to follow best practices for the development of scientific software. For instance, the default output object includes the inputs and the settings of the solver in order to ease the reproducibility of the results. However, the present version is a step back with respect to Nemoh regarding the durability of the code. Indeed, the original code is a single Fortran project with no dependency. Thus it requires far less maintenance than the present code which is built on top of several layers of fast evolving languages and software.

This more modular version of the code allows the experimentation of new techniques to improve its performance. Hierarchical matrices have been implemented in the code and their combination with the use of local symmetries of the mesh is being tested (Ancellin & Dias, 2018, 2019). It might have been more efficient to try to merge the code into an existing BEM solver instead of redeveloping advanced BEM techniques. The present work is nonetheless a step towards a fully modular code, in which the current Green's function evaluation routines could be used in another BEM solver (e.g. (Alouges & Aussal, 2018)), and conversely other methods for the computation of the Green's function (Xie et al., 2018) could be plugged in to the present user interface.

Acknowledgment

This work is funded by Science Foundation Ireland (SFI) under Marine Renewable Energy Ireland (MaREI), the SFI Centre for Marine Renewable Energy Research (grant 12/RC/2302).

References

Alouges, F., & Aussal, M. (2018). FEM and BEM simulations with the Gypsilab framework. *SMAI-Journal of computational mathematics*, *4*, 297–318. doi:10.5802/smai-jcm.36

Ancellin, M., & Dias, F. (2018). Using the floating body symmetries to speed up the numerical computation of hydrodynamics coefficients with Nemoh. In *ASME 2018 37th International Conference on Ocean, Offshore and Arctic Engineering (OMAE2018)* (p. V009T13A031). doi:10.1115/OMAE2018-77924

Ancellin, M., & Dias, F. (2019). Hierarchical Toeplitz matrices for the linear potential flow simulation of wave energy converters with local regularities. *in preparation*.

Babarit, A. (2017). *Ocean wave energy conversion: Resource, technologies and performance*. Elsevier.

Babarit, A., & Delhommeau, G. (2015). Theoretical and numerical aspects of the open source BEM solver NEMOH. In *Proceedings of the eleventh European Wave and Tidal Energy Conference (EWTEC2015)*.

Delhommeau, G. (1987). *Problèmes de diffraction-radiation et de résistance des vagues : Étude théorique et résolution numérique par la méthode des singularités* (PhD thesis). École Nationale Supérieure de Mécanique de Nantes, Nantes.

Delhommeau, G. (1989). Amélioration des codes de calcul de diffraction-radiation au premier ordre. In *Actes des 2émes journées de l'Hydrodynamique*.

Delhommeau, G. (1993). The seakeeping codes Aquadyn and Aquaplus. In *19th WEGEMT school, numerical simulation of hydrodynamics: Ships and offshore structures*. Presented at the 19th WEGEMT school, numerical simulation of hydrodynamics: Ships and offshore structures.

Folley, M. (2016). *Numerical modelling of wave energy converters: State-of-the-art techniques for single devices and arrays*. Academic Press.

Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, *5*(1). doi:10.5334/jors.148

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, *9*(3), 90–95. doi:10.1109/MCSE.2007.55

Jones, E., Oliphant, T., Peterson, P., & others. (2001). SciPy: Open source scientific tools for Python. Retrieved from http://www.scipy.org/

Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., & Bruhin, F. (2004). Pytest 4.3. Retrieved from https://github.com/pytest-dev/pytest

Newman, J. (1977). *Marine hydrodynamics*. MIT Press.

Oliphant, T. (2006). *NumPy: A guide to NumPy*. Trelgol Publishing. Retrieved from http://www.numpy.org/

Penalba Retes, M., Kelly, T., & Ringwood, J. (2017). Using NEMOH for modelling wave energy converters: A comparative study with WAMIT. In *Proceedings of the twelfth European Wave and Tidal Energy Conference (EWTEC2017)* (pp. 631–1–631–10).

Peterson, P. (2009). F2PY: A tool for connecting fortran and python programs. *Int. J. Comput. Sci. Eng.*, *4*(4), 296–305. doi:10.1504/IJCSE.2009.029165

Rongère, F. (2017). Meshmagick. Retrieved from https://github.com/LHEEA/meshmagick

Schroeder, W., Martin, K., & Lorensen, B. (2006). *The visualization toolkit (4th ed.)*. Kitware.

Xie, C., Choi, Y., Rongère, F., Clément, A., Delhommeau, G., & Babarit, A. (2018). Comparison of existing methods for the calculation of the infinite water depth free-surface green function for the wave–structure interaction problem. *Applied Ocean Research*, *81*, 150–163. doi:10.1016/j.apor.2018.10.007