# WaterLily.jl: Simulating fluid flow and dynamic geometries with heterogeneous execution Julia solver

First Author[a,*], Second Author[a,b], Third Author[b]

[a] *First Address*
[b] *Second Address*

**Abstract**

Integrating computational fluid dynamics (CFD) software into optimization and machine-learning frameworks is hampered by the rigidity of classic computational languages and the slow performance of more flexible high-level languages. WaterLily.jl is an open-source incompressible viscous flow solver written in the Julia language. The small code base is multidimensional, multi-platform and backend-agnostic (serial CPU, multithreaded, & GPU execution). The computational time per time step scales linearly with the number of degrees of freedom on CPUs, and we see up to a 182x speed-up using CUDA kernels. This leads to comparable performance with Fortran solvers on many research-scale problems opening up exciting possible future applications on the cutting edge of machine-learning research.

*Keywords:* heterogeneous programming; Cartesian-grid methods; Julia; GPU

## PROGRAM SUMMARY

*Program Title:* WaterLily.jl
*CPC Library link to program files:* (to be added by Technical Editor)
*Developer's repository link:* https://github.com/weymouth/WaterLily.jl
*Code Ocean capsule:* (to be added by Technical Editor)
*Licensing provisions(please choose one):* MIT
*Programming language:* Julia
*Supplementary material:*

[*]Corresponding author.
*E-mail address:* firstAuthor@somewhere.edu

*Nature of problem(approx. 50-250 words):*
*Solution method(approx. 50-250 words):*
*Additional comments including restrictions and unusual features (approx. 50-250 words):*

## References

[1] Program summary reference 1

[2] Program summary reference 2

[3] Program summary reference 3

## 1. Introduction

Similar to the arXiv paper, and mention some other codes here too. Some references published at CPC that we could cite and follow to organise this paper:

- [1]
- [2, 3]
- [4]

## 2. Numerical methods

- FV, BDIM, GMG

## 3. Software design

- N-dimensional solver
- KernelAbstractions.jl
- Auto bodies

## 4. Benchmark and validation

- Cases: TGV (no body), fixed sphere, transverse motion cylinder
- Architectures: NVIDIA H100 (Marenostrum 5 at BSC), AMD Radeon Instinct MI50 (CTE-AMD cluster at BSC), serial execution (?)

Try different grid sizes, testing the GPUs at different capacity.
Profiling with Nsight (see [4]) for kernel time, CPU-GPU communication.

## 5. Flashy test cases

For example...

- 2D pitching/heaving airfoils (parametric bodies)
- Jellyfish (expanding/contracting bodies)
- Adaptive sampling (CFD+"ML"): what parametric space do we sample? Maybe shark test case finding a surface response for thrust or vorticity field.

## 6. Conclusions

## References

[1] J. Romero, P. Costa, M. Fatica, Distributed-memory simulations of turbulent flows on modern GPU systems using an adaptive pencil decomposition library, in: Proceedings of the Platform for Advanced Scientific Computing Conference, ACM, 2022. doi:10.1145/3539781.3539797.

[2] M. Bernardini, D. Modesti, F. Salvadore, S. Pirozzoli, Streams: A high-fidelity accelerated solver for direct numerical simulation of compressible turbulent flows, Computer Physics Communications 263 (2021) 107906. doi:https://doi.org/10.1016/j.cpc.2021.107906.

[3] M. Bernardini, D. Modesti, F. Salvadore, S. Sathyanarayana, G. D. Posta, S. Pirozzoli, STREAmS-2.0: Supersonic turbulent accelerated navier-stokes solver version 2.0, Computer Physics Communications 285 (2023) 108644. doi:10.1016/j.cpc.2022.108644.

[4] L. Gasparino, F. Spiga, O. Lehmkuhl, SOD2D: A GPU-enabled spectral finite elements method for compressible scale-resolving simulations, Computer Physics Communications xxx (2023) xxx. doi:xxx.