The Journal of Open Source Software

PetIBM: toolbox and applications of the immersed-boundary method on distributed-memory architectures

Pi-Yueh Chuang[1], **Olivier Mesnard**[1], **Anush Krishnan**[2], **and Lorena A. Barba**[1]

1 The George Washington University **2** nuTonomy Inc. (previously at Boston University)

DOI: 10.21105/joss.00558

Software

- Review
- Repository
- Archive

Submitted: 26 January 2018
Published: 09 May 2018

Summary

PetIBM solves the two- and three-dimensional Navier-Stokes equations with an immersed-boundary method on fixed structured Cartesian grids. In this method, a collection of Lagrangian markers defines the immersed boundary (where boundary conditions are enforced) and the fluid equations are solved over the extended domain (including the body domain). The Eulerian mesh remains unmodified when computing the flow around multiple moving immersed bodies, which removes the need for remeshing at every time step. PetIBM discretizes the fluid equations using a second-order finite-difference scheme, various optional time-integrators, and a fully discrete projection method (Perot (1993)). It implements two immersed-boundary algorithms: the immersed-boundary projection method (Taira and Colonius (2007)) and its decoupled version (Li et al. (2016)). Other open-source software packages offer immersed-boundary solvers: for example, IBAMR (Griffith et al. (2007), Bhalla et al. (2013)) is a long-standing C++ library with MPI parallelization that also provides adaptive mesh refinement. It can handle deforming immersed bodies and has been used in a variety of scenarios, including cardiac fluid dynamics, swimming, insect flight, and others. PetIBM and IBAMR use different immersed-boundary schemes, however. We developed PetIBM to work with the immersed-boundary projection method, which is based on the fully discrete formulation of Perot and thus eliminates the need for pressure boundary conditions (which have caused many headaches for CFD practitioners!). PetIBM features an operator-based design, so it can be used as a toolbox for researching new solution methods. It is also capable of using GPU architectures, a feature missing from other software, as far as we know. A previous project implementing immersed-boundary methods on GPU architecture is cuIBM (Krishnan, Mesnard, and Barba (2017)), but it is limited to two-dimensional problems that fit on a single GPU device.

PetIBM is written in C++ and relies on the PETSc library (Balay et al. (1997), Balay et al. (2017)) to run on memory-distributed architectures. PetIBM can solve one or several linear systems on multiple distributed CUDA-capable GPU devices with the NVIDIA AmgX library and AmgXWrapper (Chuang and Barba (2017)).

PetIBM has already been used to generate results published in Mesnard and Barba (2017), a full replication of a study on the aerodynamics of a gliding snake species (Krishnan et al. (2014)). PetIBM is currently used to compute the three-dimensional flow of a gliding-snake model on the cloud platform Microsoft Azure.

References

Balay, Satish, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, et al. 2017. "PETSc Users Manual." ANL-95/11 - Revision 3.8. Argonne National Laboratory.

Balay, Satish, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. 1997. "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries." In *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, 163–202. Birkhäuser Press.

Bhalla, Amneet Pal Singh, Rahul Bale, Boyce E Griffith, and Neelesh A Patankar. 2013. "A Unified Mathematical Framework and an Adaptive Numerical Method for Fluid–Structure Interaction with Rigid, Deforming, and Elastic Bodies." *Journal of Computational Physics* 250. Elsevier:446–76. https://doi.org/10.1016/j.jcp.2013.04.033.

Chuang, Pi-Yueh, and Lorena A. Barba. 2017. "AmgXWrapper: An Interface Between PETSc and the NVIDIA AmgX Library." *The Journal of Open Source Software* 2 (16). The Open Journal:280. https://doi.org/10.21105/joss.00280.

Griffith, Boyce E, Richard D Hornung, David M McQueen, and Charles S Peskin. 2007. "An Adaptive, Formally Second Order Accurate Version of the Immersed Boundary Method." *Journal of Computational Physics* 223 (1). Elsevier:10–49. https://doi.org/10.1016/j.jcp.2006.08.019.

Krishnan, Anush, Olivier Mesnard, and Lorena A. Barba. 2017. "cuIBM: A GPU-Based Immersed Boundary Method Code." *The Journal of Open Source Software* 2 (15). The Open Journal:301. https://doi.org/10.21105/joss.00301.

Krishnan, Anush, John J Socha, Pavlos P Vlachos, and LA Barba. 2014. "Lift and Wakes of Flying Snakes." *Physics of Fluids* 26 (3). AIP:031901. https://doi.org/10.1063/1.4866444.

Li, Ru-Yang, Chun-Mei Xie, Wei-Xi Huang, and Chun-Xiao Xu. 2016. "An Efficient Immersed Boundary Projection Method for Flow over Complex/Moving Boundaries." *Computers & Fluids* 140. Elsevier:122–35. https://doi.org/10.1016/j.compfluid.2016.09.017.

Mesnard, Olivier, and Lorena A Barba. 2017. "Reproducible and Replicable Computational Fluid Dynamics: It's Harder Than You Think." *Computing in Science & Engineering* 19 (4). IEEE:44–55. https://doi.org/10.1109/MCSE.2017.3151254.

Perot, J Blair. 1993. "An Analysis of the Fractional Step Method." *Journal of Computational Physics* 108 (1). Elsevier:51–58. https://doi.org/10.1006/jcph.1993.1162.

Taira, Kunihiko, and Tim Colonius. 2007. "The Immersed Boundary Method: A Projection Approach." *Journal of Computational Physics* 225 (2). Elsevier:2118–37. https://doi.org/10.1016/j.jcp.2007.03.005.