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 Department of Mechanical and Aerospace Engineering, The George Washington University, Washington, DC, USA **2** nuTonomy Inc., Cambridge, MA, USA (previously at Boston University)

DOI: 10.21105/joss.00558

Software

- Review
- Repository
- Archive

Submitted: 26 January 2018
Published: 24 May 2018

Summary

PetIBM is a C++ library with ready-to-use application codes to solve the two- and three-dimensional incompressible Navier-Stokes equations on fixed structured Cartesian grids with an immersed-boundary method (IBM). PetIBM runs on distributed-memory architectures and can be used to compute the flow around multiple moving rigid immersed boundaries (with prescribed kinematics).

In the IBM framework, 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 on staggered grids and thus eliminates the need for pressure boundary conditions, which have caused many headaches for CFD practitioners (Gresho and Sani (1987), Sani and Gresho (1994)). PetIBM features an operator-based design, providing routines to create and manipulate discrete operators (e.g., gradient, divergence, Laplacian, convection, diffusion, etc.), so it can be used as a toolbox for researching new solution methods. It is also capable of using graphics processing unit (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)) for data structures and parallel routines to run on memory-distributed archi-

tectures. 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)). The software package includes extended documentation as well as many examples to guide users.

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.

Appendix: mathematical formulation

PetIBM solves the Navier-Stokes equations on an extended discretization grid that includes the interior of the immersed boundary. To model the presence of the boundary, a forcing term is added to the momentum equation and an additional equation for the no-slip condition completes the system. Many variants of the immersed-boundary method (IBM) depend on one models the forcing. In PetIBM, we use regularized-delta functions to transfer data between the Eulerian grid and the Lagrangian boundary points. The system of equation is:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} + \int_s \mathbf{f}\left(\xi\left(s,t\right)\right) \delta\left(\xi - \mathbf{x}\right) ds \\ \nabla \cdot \mathbf{u} = 0 \\ \mathbf{u}\left(\xi\left(s,t\right)\right) = \int_{\mathbf{x}} \mathbf{u}\left(\mathbf{x}\right) \delta\left(\mathbf{x} - \xi\right) d\mathbf{x} \end{cases} \tag{1}$$

where $\mathbf{u}$ is the velocity field, p is the fluid pressure, and Re is the Reynolds number.

Currently, PetIBM provides two application codes implementing different versions of the IBM: (1) an immersed-boundary projection method (IBPM) based on the work of Taira and Colonius (2007) and (2) a decoupled version of the IBPM proposed by Li et al. (2016). Those two methods fit into the framework of the projection approach of Perot (1993). The equations are fully discretized (space and time) to form an algebraic system to be solved for the velocity u^{n+1}, the pressure field ϕ, and the Lagrangian forces $\tilde{f}$. The discretized system is:

$$\begin{bmatrix} A & G & H \\ D & 0 & 0 \\ E & 0 & 0 \end{bmatrix} \begin{pmatrix} u^{n+1} \\ \phi \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} r^n \\ 0 \\ u_B^{n+1} \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \\ 0 \end{pmatrix} \tag{2}$$

where D, G, and A are the divergence, gradient, and implicit operators, respectively. E and H are the interpolation and spreading operators, respectively, used to transfer the data between the Eulerian grid and the Lagrangian boundary points. On the right-hand side, r^n gathers all the explicit terms and u_B^{n+1} is the known (prescribed) boundary velocity; bc_1 and bc_2 contain the boundary terms that arise from the discretization of momentum and continuity equations, respectively.

In the IBPM, we solve a modified Poisson system for the pressure field and Lagrangian forces, coupled together. This way, the divergence-free condition and no-slip constraint are simultaneously enforced on the velocity field at the end of the time step. The fully discretized system can be cast into the following:

$$\begin{bmatrix} A & Q_2 \\ Q_1 & 0 \end{bmatrix} \begin{pmatrix} u^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \tag{3}$$

with

$$Q_1 \equiv \begin{bmatrix} D \\ E \end{bmatrix};\ Q_2 \equiv [G, H];\ \lambda \equiv \begin{pmatrix} \phi \\ \tilde{f} \end{pmatrix};\ r_1 \equiv r^n + bc_1;\ r_2 \equiv \begin{pmatrix} bc_2 \\ u_B^{n+1} \end{pmatrix}$$

In practice, we never form the full system. Instead, we apply a block-LU decomposition as follow:

$$\begin{bmatrix} A & 0 \\ Q_1 & -Q_1 A^{-1} Q_2 \end{bmatrix} \begin{bmatrix} I & A^{-1}Q_2 \\ 0 & I \end{bmatrix} \begin{pmatrix} u^{n+1} \\ \lambda \end{pmatrix} = \begin{bmatrix} A & 0 \\ Q_1 & -Q_1 A^{-1} Q_2 \end{bmatrix} \begin{pmatrix} u^* \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \tag{4}$$

Thus, we retrieve the sequence of operations of the traditional projection method. We solve a system for an intermediate velocity field that is corrected, after solving a modified Poisson system for the variable λ, to enforce the divergence-free condition and the no-slip constraint at the location of the immersed boundary. The sequence is:

$$Au^* = r_1 \tag{5}$$
$$Q_1 A^{-1} Q_2 \lambda = Q_1 u^* - r_2 \tag{6}$$
$$u^{n+1} = u^* - A^{-1} Q_1 \lambda \tag{7}$$

The IBPM implemented in PetIBM solves, at every time step, Equations (5) to (6). (Note: the inverse of the implicit operator A^{-1} is approximated by a finite Taylor series expansion.)

The IBPM requires solving, at each time step, an expensive modified Poisson system, $Q_1 A^{-1} Q_2$, whose non-zero structure changes when the location of the immersed boundary is moving. In the PetIBM implementation of the decoupled IBPM, we apply a second block-LU decomposition to decouple the pressure field from the Lagrangian forces and recover a classical Poisson system. The fully discretized algebraic system can be cast into:

$$\begin{bmatrix} A & H & G \\ E & 0 & 0 \\ D & 0 & 0 \end{bmatrix} \begin{pmatrix} u^{n+1} \\ \tilde{f} \\ \phi \end{pmatrix} = \begin{pmatrix} r^n \\ u_B^{n+1} \\ 0 \end{pmatrix} + \begin{pmatrix} bc_1 \\ 0 \\ bc_2 \end{pmatrix} \tag{8}$$

The velocity u^{n+1} and the Lagrangian forces $\tilde{f}$ are coupled together to form a new unknown γ^{n+1}, as follows:

$$\begin{bmatrix} \bar{A} & \bar{G} \\ \bar{D} & 0 \end{bmatrix} \begin{pmatrix} \gamma^{n+1} \\ \phi \end{pmatrix} = \begin{pmatrix} \bar{r_1} \\ \bar{r_2} \end{pmatrix} \tag{9}$$

where

$$\bar{A} \equiv \begin{bmatrix} A & H \\ E & 0 \end{bmatrix};\ \bar{G} \equiv \begin{bmatrix} G \\ 0 \end{bmatrix};\ \bar{D} \equiv \begin{bmatrix} D & 0 \end{bmatrix}$$

and

$$\gamma^{n+1} \equiv \begin{pmatrix} u^{n+1} \\ \tilde{f} \end{pmatrix};\ \bar{r_1} \equiv \begin{pmatrix} r_n + bc_1 \\ u_B^{n+1} \end{pmatrix};\ \bar{r_2} \equiv bc_2$$

Two successive block-LU decompositions are applied to decouple the Lagrangian forces $\tilde{f}$ from γ^{n+1} and to decouple the velocity from the pressure field.

The first block-LU decomposition decouples the pressure field from the new unknown γ^{n+1}:

$$\begin{bmatrix} \bar{A} & 0 \\ \bar{D} & -\bar{D}\bar{A}^{-1}\bar{G} \end{bmatrix} \begin{bmatrix} I & \bar{A}^{-1}\bar{G} \\ 0 & I \end{bmatrix} \begin{pmatrix} \gamma^{n+1} \\ \phi \end{pmatrix} = \begin{bmatrix} \bar{A} & 0 \\ \bar{D} & -\bar{D}\bar{A}^{-1}\bar{G} \end{bmatrix} \begin{pmatrix} \gamma^{*} \\ \phi \end{pmatrix} = \begin{pmatrix} \bar{r_1} \\ \bar{r_2} \end{pmatrix} \tag{10}$$

which leads to the following sequence of operations:

$$\bar{A}\gamma^{*} = \bar{r_1} \tag{11}$$
$$\bar{D}\bar{A}^{-1}\bar{G}\phi = \bar{D}\gamma^{*} - \bar{r_2} \tag{12}$$
$$\gamma^{n+1} = \gamma^{*} - \bar{A}^{-1}\bar{G}\phi \tag{13}$$

A second block-LU decomposition is applied to the first equation above:

$$\begin{bmatrix} A & 0 \\ E & -EA^{-1}H \end{bmatrix} \begin{bmatrix} I & A^{-1}H \\ 0 & I \end{bmatrix} \begin{pmatrix} u^{*} \\ \tilde{f} \end{pmatrix} = \begin{bmatrix} A & 0 \\ E & -EA^{-1}H \end{bmatrix} \begin{pmatrix} u^{**} \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} r^{n} + bc_1 \\ u_B^{n+1} \end{pmatrix} \tag{14}$$

and we end up with the following sequence:

$$Au^{**} = r^{n} + bc_1 \tag{15}$$
$$EA^{-1}H\tilde{f} = Eu^{**} - u_B^{n+1} \tag{16}$$
$$u^{*} = u^{**} - A^{-1}H\tilde{f} \tag{17}$$

The decoupled version of the IBPM implemented in PetIBM solves, at every time step, Equations (15) to (17) followed by Equations (12) and (13).

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.

Gresho, Philip M, and Robert L Sani. 1987. "On Pressure Boundary Conditions for the Incompressible Navier-Stokes Equations." *International Journal for Numerical Methods in Fluids* 7 (10). Wiley Online Library:1111–45. https://doi.org/10.1002/fld.1650071008.

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.

Sani, Robert L, and Philip M Gresho. 1994. "Résumé and Remarks on the Open Boundary Condition Minisymposium." *International Journal for Numerical Methods in Fluids* 18 (10). Wiley Online Library:983–1008. https://doi.org/10.1002/fld.1650181006.

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.