Validation of the cuIBM code for Navier-Stokes equations with immersed boundary methods [1]

Anush Krishnan, Lorena A. Barba

6 July 2012

We have developed a Navier-Stokes solver, called `cuIBM`, to simulate incompressible flows using immersed boundary methods. This document provides background on the numerical methods implemented in the `cuIBM` code framework and evidence of the validation exercise carried out by the authors. The code provides a growing set of options for flows with immersed boundaries, is written in `C++` and uses GPU hardware via `Cuda` kernels and calls to the `Cusp` library. The validation tests in two-dimensions use an analytical solution (Couette flow) and several experimental results to contrast with the numerical solutions. The experimental benchmarks include a lid-driven cavity at $Re = 100$, impulsively started flow around a circular cylinder at low and moderate Reynolds numbers, cylinder wake flow, and flow over both heaving and flapping airfoils.

Numerical Method

Brief overview of the immersed boundary method

The first immersed boundary method (IBM) was introduced by Peskin for the application of computational fluid dynamics to the simulation of flow in heart valves.[2] His method added a forcing term in the Navier-Stokes equation to account for the presence of a moving boundary, modeling the body as a collection of springs. Boundary points are placed at the equilibrium positions of the springs and are allowed to move with the flow, and the force is calculated using Hooke's law. The force is then transferred from the solid to the fluid grid by some kind of interpolation. The defining feature of the IBM is that the computational grid in the fluid domain does not conform to the geometry of the immersed body. Thus, it allows simulating flows with moving boundaries without the need of constant re-meshing.

[2] C. S. Peskin. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.*, 10(2):252–271, 1972. ISSN 0021-9991. URL `http://linkinghub.elsevier.com/retrieve/pii/0021999172900654`

The IBM approach was later extended to treat rigid bodies, and since then several authors have proposed variants that differ in the way of obtaining the boundary force. Mittal and Iaccarino give an excellent overview of the various IBM techniques introduced until 2005.[3] These are generally easy to implement, but have some drawbacks: most do not directly apply the no-slip boundary condition, and they often require several ad hoc parameters to be chosen by the users carefully; many of them also have stringent requirements on the time step size.

[3] R. Mittal and G. Iaccarino. Immersed boundary methods. *Ann. Rev. Fluid Mech.*, 37(1):239–261, 2005. ISSN 0066-4189. DOI: 10.1146/annurev.fluid.37.061903.175743. URL `http://arjournals.annualreviews.org/doi/abs/10.1146%2Fannurev.fluid.37.061903.175743`

In the `cuIBM` code, we aim to provide options to test several IBM techniques, but in the first instance we only implemented the method proposed by Taira and Colonius.[4] This formulation of the IBM is a projection method for the Navier-Stokes equations with the forcing term that uses the matrix factorization approach of Perot.[5] In the general projection method for incompressible Navier-Stokes, the pressure acts as a Lagrange multiplier to ensure the zero-divergence condition on velocity. Similarly, Taira and Colonius consider the body forcing term as a Lagrange multiplier that ensures the no-slip condition on the solid boundary. During the projection step, the velocity field is updated so that both of these constraints are satisfied. They also use an implicit scheme for diffusion, allowing relatively large time steps, with CFL numbers as high as 1. One shortcomings of this method is the requirement that boundary-points be separated by distances that are nearly equal to the grid spacing.

[4] K. Taira and T. Colonius. The immersed boundary method: A projection approach. *J. Comput. Phys.*, 225(2): 2118–2137, 2007. ISSN 00219991. DOI: 10.1016/j.jcp.2007.03.005. URL `http://linkinghub.elsevier.com/retrieve/pii/S0021999107001234`

[5] J. B. Perot. An analysis of the fractional step method. *J. Comput. Phys.*, 108(1):51–58, 1993. ISSN 0021-9991. URL `http://www.ecs.umass.edu/mie/faculty/perot/Papers/FSM0.pdf`

Mathematical formulation of the IBM used in this work

The governing equations are:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}\mathbf{u} = -\boldsymbol{\nabla} p + \nu\Delta\mathbf{u} + \int_s \mathbf{f}(\boldsymbol{\xi}(s,t))\delta(\boldsymbol{\xi}-\mathbf{x})ds \tag{1a}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{1b}$$

$$\begin{aligned}\mathbf{u}(\boldsymbol{\xi}(s,t)) &= \int_s \mathbf{u}(\mathbf{x})\delta(\mathbf{x}-\boldsymbol{\xi})d\mathbf{x} \\ &= \mathbf{u}_B(\boldsymbol{\xi}(s,t)),\end{aligned} \tag{1c}$$

where $\mathbf{f}$ is a singular force distribution along the solid boundary and $\mathbf{u}_B$ is the velocity of the body. After discretization, we obtain the following set of algebraic equations:

$$\begin{aligned}\hat{A}u^{n+1} - \hat{r}^n &= -\hat{G}\phi + \hat{bc}_1 + \hat{H}f \\ \hat{D}u^{n+1} &= bc_2 \\ \hat{E}u^{n+1} &= u_B^{n+1},\end{aligned} \tag{2}$$

The velocity at the current time step u^n is known and gives the values of the convective terms $\hat{r}^n$; $\hat{bc}_1$ and bc_2 are the boundary conditions on the velocity. $\hat{G}$ and $\hat{D}$ are discrete gradient and divergence operators. $\hat{H}$ and $\hat{E}$ are the regularization and interpolation matrices, used to transfer values of the flow variables between the Eulerian fluid grid and the Lagrangian body grid.

which is written in matrix form as

$$\begin{pmatrix} \hat{A} & \hat{G} & \hat{H} \\ \hat{D} & 0 & 0 \\ \hat{E} & 0 & 0 \end{pmatrix} \begin{pmatrix} u^{n+1} \\ \phi \\ f \end{pmatrix} = \begin{pmatrix} \hat{r}^n \\ 0 \\ u_B^{n+1} \end{pmatrix} + \begin{pmatrix} \hat{bc}_1 \\ -bc_2 \\ 0 \end{pmatrix}. \tag{3}$$

Here, ϕ is a vector of pressure values and f is a vector of force values at the boundary points. The interpolation in (**??**) can be discretized as:

$$u_k = \sum_i u_i d(x_i - \xi_k) d(y_i - \eta_k)\Delta x\Delta y, \tag{4}$$

The velocity u_k at a point (ξ_k, η_k) on the immersed boundary is calculated by convolving the velocities u_i at points (x_i, y_i) on the Eulerian fluid grid with a discrete two-dimensional delta function.

The discrete delta function is a product of smoothed one-dimensional delta functions $d_h(r)$ along each Cartesian direction. We choose the following, where h is the cell width,[6]

[6] A. M. Roma, C. S. Peskin, and M. J. Berger. An adaptive version of the immersed boundary method. *J. Comput. Phys.*, 153(2):509–534, 1999

$$d_h(r) = \begin{cases} \frac{1}{6h}\left(5 - 3\frac{|r|}{h} - \sqrt{-3\left(1-\frac{|r|}{h}\right)^2 + 1}\right), & 0.5 < \frac{|r|}{h} \leq 1.5 \\ \frac{1}{3h}\left(1 + \sqrt{-3\left(\frac{|r|}{h}\right)^2 + 1}\right), & 0 < \frac{|r|}{h} \leq 0.5 \\ 0, \quad \text{otherwise.} \end{cases}$$

From the above, we obtain the elements of matrix $\hat{E}$,

$$\hat{E}_{k,i} = \Delta x \Delta y \, d(x_i - \xi_k) \, d(y_i - \eta_k). \tag{5}$$

Use of the discrete delta function requires the grid to be uniform near the immersed boundary. Sufficient number of boundary points need to be chosen to prevent flow leakage, and the spacing between boundary points should be nearly equal to the cell spacing of the Eulerian fluid grid.

The matrix $\hat{H}$ is obtained similarly, discretizing the forcing term in (**??**). Appropriate transformations result in a system for the momentum fluxes at cell boundaries, q^{n+1}, and modified forces $\tilde{f}$, which can be written as:

$$\begin{pmatrix} A & Q \\ Q^T & 0 \end{pmatrix} \begin{pmatrix} q^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \tag{6}$$

The transformation makes the left-hand-side matrix symmetric and we drop the hats from the symbols representing the transformed sub-matrices.

where

$$Q = \begin{bmatrix} G & E^T \end{bmatrix}, \quad \lambda = \begin{pmatrix} \phi \\ \tilde{f} \end{pmatrix}, \quad r_1 = r^n + bc_1, \quad r_2 = \begin{pmatrix} -bc_2 \\ u_B^{n+1} \end{pmatrix} \tag{7}$$

Perot's approximate factorization results in the following set of equations, giving velocity field at time step $n+1$:

$$Aq^* = r_1 \tag{8a}$$

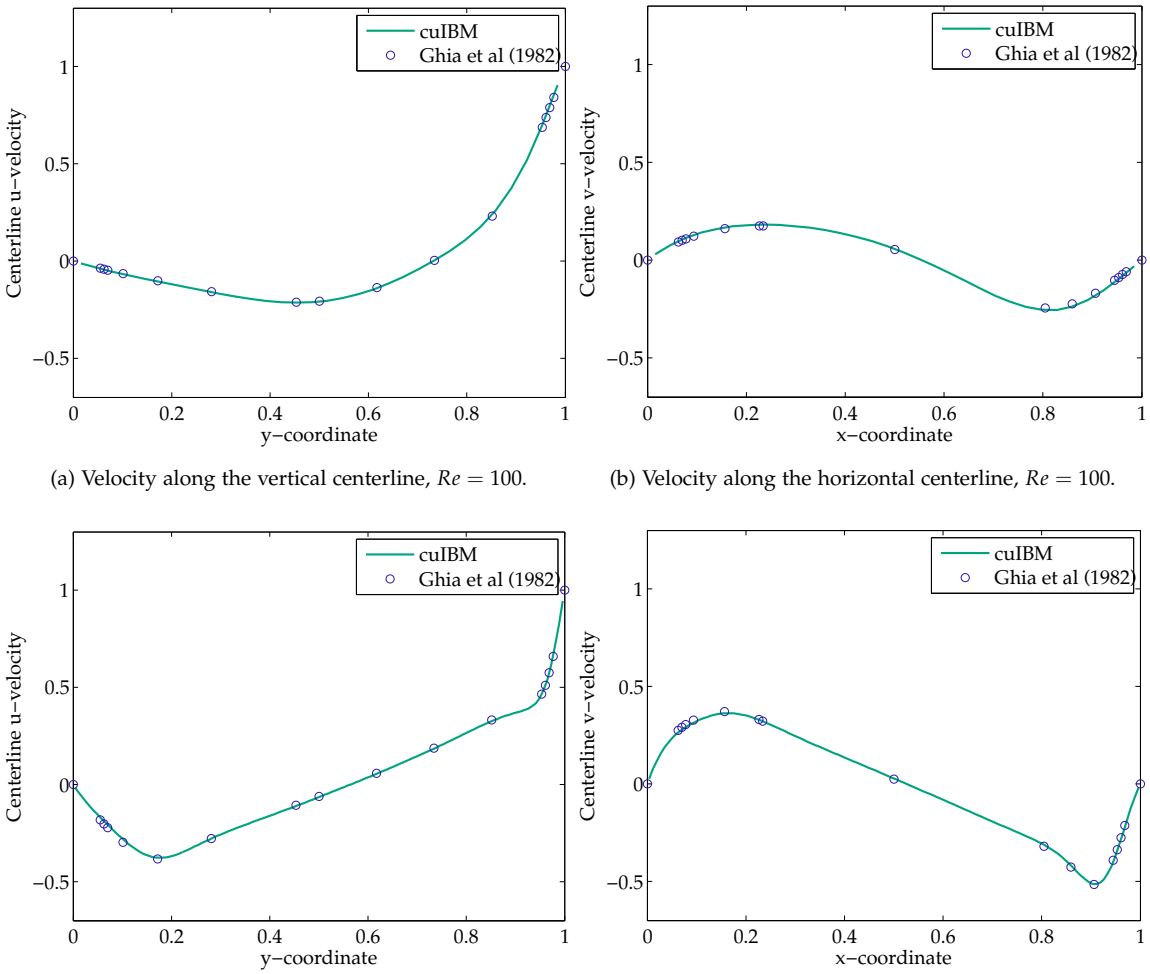
$$Q^T B^N Q\lambda = Q^T q^* - r_2 \tag{8b}$$

$$q^{n+1} = q^* - B^N Q\lambda \tag{8c}$$

Here, q^* is the intermediate solution for the fluxes, used in the projection step, and B^N is an N^{th}-order approximation of A^{-1}.

Numerical schemes

The `cuIBM` code uses a staggered grid, with pressure defined at the center of each grid cell, and velocity fluxes defined on the cell faces. The convection terms are discretized using a symmetric, conservative finite-difference scheme with explicit second-order Adams-Bashforth time stepping. Diffusion terms are discretized using central differences and advanced in time with the Crank-Nicolson scheme. The pressure and body forces are calculated implicitly. Note that no explicit boundary conditions need to be specified for the pressure or the intermediate velocity, which is an advantage of Perot's matrix factorization approach. The above system of equations is solved to obtain the velocity field at time step $n+1$, the pressure (to a constant) and the body forces. For the two systems of algebraic equations, (**??**) and (**??**), we use a conjugate gradient solver.



(a) Velocity along the vertical centerline, $Re = 100$.

(b) Velocity along the horizontal centerline, $Re = 100$.

(c) Velocity along the vertical centerline, $Re = 1000$.

(d) Velocity along the horizontal centerline, $Re = 1000$.

Figure 1: Validation using a lid-driven cavity.

Validation

Lid-driven cavity

One of the simplest validation tests for a two-dimensional Navier-Stokes solver is the lid-driven cavity: the flow in a square cavity, with the top wall of the cavity (the lid) moving horizontally with constant velocity. This benchmark does not make use of the immersed boundary method but it is a test for the underlying Navier-Stokes solver. The Reynolds number is calculated using the speed of the lid, the length of the side of the square cavity and the kinematic viscosity of the fluid. The numerical solution was advanced in time until reaching steady state and the velocity is compared with the results by Ghia et al.[7] See Figure **??**. We used uniform grids, 32×32 for the $Re = 100$ case and 128×128 for the the $Re = 1000$ case.

[7] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comput. Phys.*, 48(3): 387–411, 1982. DOI: 10.1016/0021-9991(82)90058-4. URL `http://www.sciencedirect.com/science/article/pii/0021999182900584`

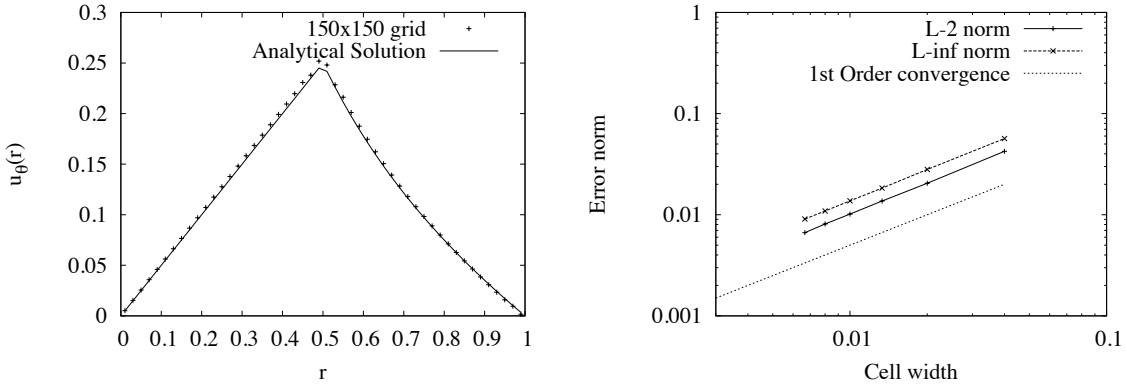
Couette flow between concentric cylinders

This is a simple validation test using an analytical solution, with the immersed boundary method in internal-flow mode. We calculate the flow between two concentric cylinders of radii $r_i = 0.5$ and $r_o = 1$, centered at the origin. The outer cylinder is held stationary while the inner cylinder is impulsively rotated from rest with an angular velocity $\Omega = 0.5$. The cylinders are contained in a square, stationary domain of side 1.5, centered at the origin.

The steady-state analytical solution for this flow is known. In the interior of the inner cylinder we have solid body rotation while between the two cylinders the azimuthal velocity is given by:

$$u_\theta(r) = \Omega r_i \frac{(r_o/r - r/r_o)}{(r_o/r_i - r_i/r_o)}. \tag{9}$$

We set the kinematic viscosity to $\nu = 0.03$ and obtained the numerical solution using six different grid sizes, ranging from 75×75 to 450×450, to calculate the observed order of convergence. Figure **????** shows plots of the L^2 and L^∞ norms of the relative errors, verifying that the scheme provides first-order accuracy in space, as expected for the IBM formulation we used.



(a) Comparison of the numerical and analytical solutions.

(b) Convergence, showing errors for different grid sizes.

Figure 2: Validation using the Couette flow problem.

To verify the temporal order of convergence, we ran a simulation from $t = 0$ to $t = 8$ on a 151×151 grid, using different time steps ($\Delta t = 0.01,\ 0.005$ and 0.0025). Both first- and third-order accurate expansions of B^N were used and the calculated orders of convergence (using the L^2 norms of the differences in the solutions) at various times have been summarised in Table **??**, and are as expected.

Time	Order of convergence ($N = 1$)	Order of convergence ($N = 3$)
0.8	0.97	2.67
2	0.99	2.85
4	0.93	2.73
8	0.97	2.83

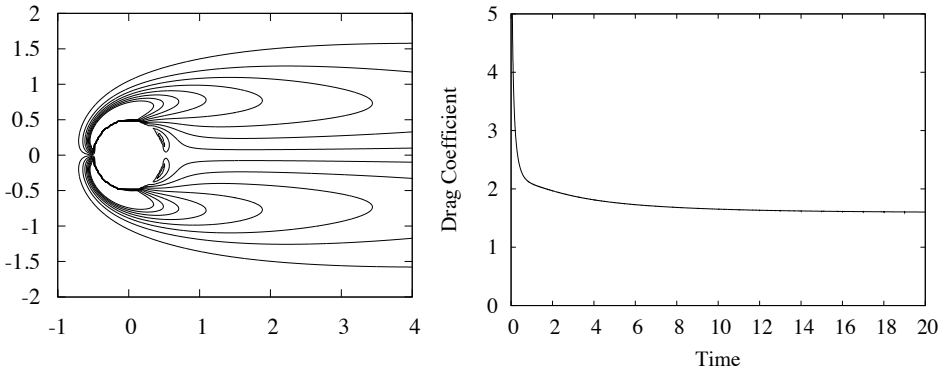
Table 1: Observed temporal order of convergence at different times for Couette-flow test.

Flow over an impulsively started cylinder

We also obtained numerical solutions of the flow over an impulsively started circular cylinder at Reynolds numbers 40, 550 and 3000, looking at the vorticity contours and also comparing the steady-state drag coefficient with experimental values published in the literature[8] and with past computations.[9] The cylinder has diameter $d = 1$, is centered at the origin and is placed in an external flow with free-stream velocity $u_\infty = 1$. The initial velocity is uniform throughout the domain, a square centered at the origin with sides of length 30. The fluid flows from left to right, and the velocity on the left, top and bottom edges was set to the free-stream velocity. A convective boundary condition ($\partial u/\partial t + u_\infty \partial u/\partial x = 0$) was used on the right edge of the domain. The minimum cell widths used near the solid boundaries were 0.02, 0.01 and 0.004 respectively for the three cases. The grid is stretched exponentially away from the body, outside a near-body region with uniform grid. See Table **??** for all grid details.

[8] D. J. Tritton. Experiments on the flow past a circular cylinder at low Reynolds numbers. *J. Fluid Mech.*, 6(04):547–567, 1959

[9] P. Koumoutsakos and A. Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *J. Fluid Mech.*, 296:1–38, 1995



(a) Steady state vorticity field.

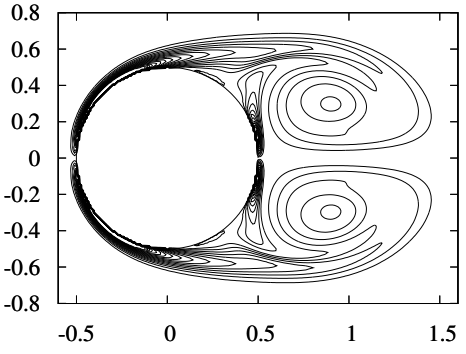
(b) Time-varying drag coefficient.

Figure 3: Validation with external flow over a circular cylinder at $Re = 40$. Contour lines in (a) are drawn from -3 to 3 in steps of 0.4.

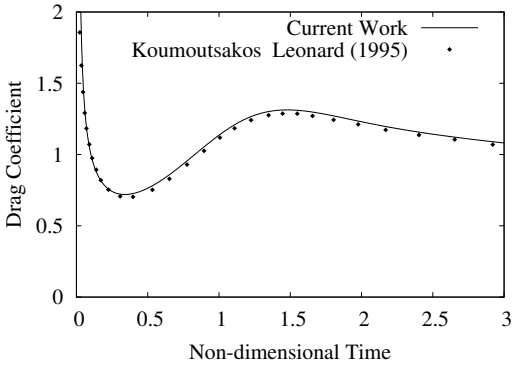
The flow at $Re = 40$ reaches a steady state with a drag coefficient of 1.57, which matches the experimental value. The vorticity fields and the unsteady drag coefficients for the cases with $Re = 550$ and 3000 also agree well with past computations of Koumoutsakos and collaborators. The results are presented in Figures **??**–**??**.

Re	$n_x \times n_y$	Δx_{min}	Extent of uniform grid	$r_{stretching}$
40	330×330	0.02	$[-0.54, 0.54]$	1.02
550	450×450	0.01	$[-0.54, 0.54]$	1.02
3000	986×986	0.004	$[-0.52, 0.52]$	1.01

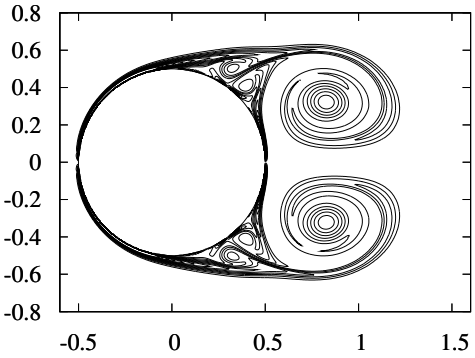
Table 2: Grid information for the impulsively started cylinder tests.



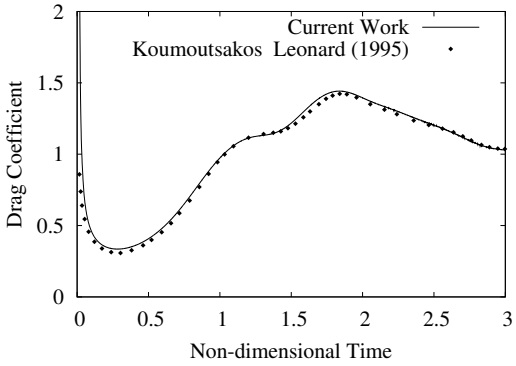
(a) Vorticity field at non-dimensional time 3.0



(b) Time-varying drag coefficient.



(c) Vorticity field at non-dimensional time 3.0



(d) Time-varying drag coefficient.

Figure 4: Validation with impulsively started flow over a circular cylinder at Reynolds 550 (top) and 3000 (bottom). The contour lines in (a) are drawn from -32 to 32 in steps of 2, and in (c) from -56 to 56 in steps of 4, both excluding the zero contour.

Unsteady flow over a circular cylinder

We obtained longer runs of flow around a circular cylinder, resulting in a von Kármán vortex street. The cylinder of diameter 1 is placed at the center of a domain of size 30×30, its initial position slightly offset in the y-direction and then brought to the center at the beginning of the run to trigger the instability in the flow and cause vortex shedding. The minimum cell width in the grids was 0.02, maintained in the wake region behind the cylinder. To the left, top and bottom of the cylinder, the grid stretches with an exponential ratio of 1.02. Table **??** lists the results obtained at different Reynolds numbers, comparing the Strouhal number of the vortex shedding with the experimental results of Williamson.[10]

[10] C. H. K. Williamson. Vortex dynamics in the cylinder wake. *Ann. Rev. Fluid Mech.*, 28:477–539, 1996

Re	C_l	C_d	St	St_{exp}
100	± 0.339	1.37 ± 0.009	0.166	0.164
150	± 0.532	1.35 ± 0.026	0.185	0.184
200	± 0.688	1.36 ± 0.042	0.197	0.197

Table 3: Computed data for flow past a circular cylinder at different Reynolds numbers and comparison with experimental results.

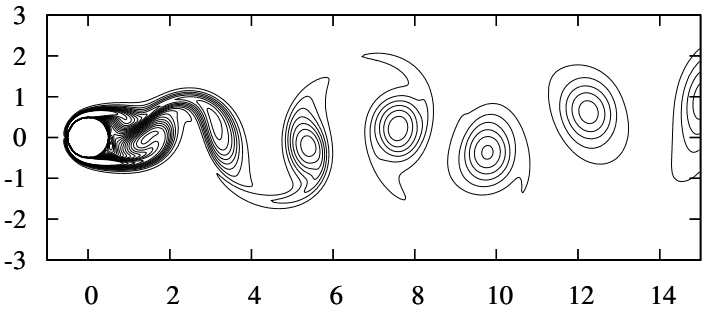


Figure 5: Flow over a circular cylinder at Reynolds 200. Vorticity contours from -5 to 5 in increments of 0.4.

THE RESULTS ABOVE validate our code with both analytical results and experimental benchmarks for flows with stationary solid interfaces. The following cases demonstrate the capability of the code to reproduce past results for flows with moving boundaries. The first case is the heaving airfoil in a uniform flow computed by Lewin and Haj-Hariri[11] and the second case is an airfoil that performs both pitching and heaving motions, simulating the flapping motion of an insect wing, as computed by Wang et al.[12]

Unsteady flow of a heaving airfoil

We computed the flow of an elliptic airfoil of thickness-to-chord ratio 0.12 and chord length $c = 1$ submerged in a free stream with velocity $u_\infty = 1.0$. The airfoil oscillates in the direction perpendicular to the free-stream velocity with frequency f and amplitude Y_{max}. The problem parameters are the reduced frequency $k = 2\pi f c/u_\infty = 2$, the non-dimensional maximum heaving velocity $kh = kY_{max}/c = 0.8$, and the Reynolds number, $Re = u_\infty c/\nu = 500$. The domain size is 30×30 and the near-body region in $[-0.52, 0.52] \times [-0.52, 0.52]$ has a uniform grid of width 0.005, and exponentially stretched grids apply in front of the airfoil (stretching ratio of 1.02), in the y-direction above and below the body (stretching ratio 1.015), and in the region $[0.52, 0.78]$ immediately behind the airfoil (stretching ratio 1.02); a uniform grid of size 0.01 follows from that region to the edge of the domain. The total size of the mesh is 1339×686 and the time step used was 0.0005. The boundary conditions are the same as those used for the earlier cases of external flow over a cylinder. The vorticity field as shown in Figure **??** matches the results in Fig. 3 of Lewin and Haj-Hariri.

[11] G. C. Lewin and H. Haj-Hariri. Modelling thrust generation of a two-dimensional heaving airfoil in a viscous flow. *J. Fluid Mech.*, 492:339–362, 2003

[12] Z. J. Wang, J. M. Birch, and M. H. Dickinson. Unsteady forces and flows in low Reynolds number hovering flight: two-dimensional computations vs. robotic wing experiments. *J. Exp. Biol.*, 207(3):449–460, 2004

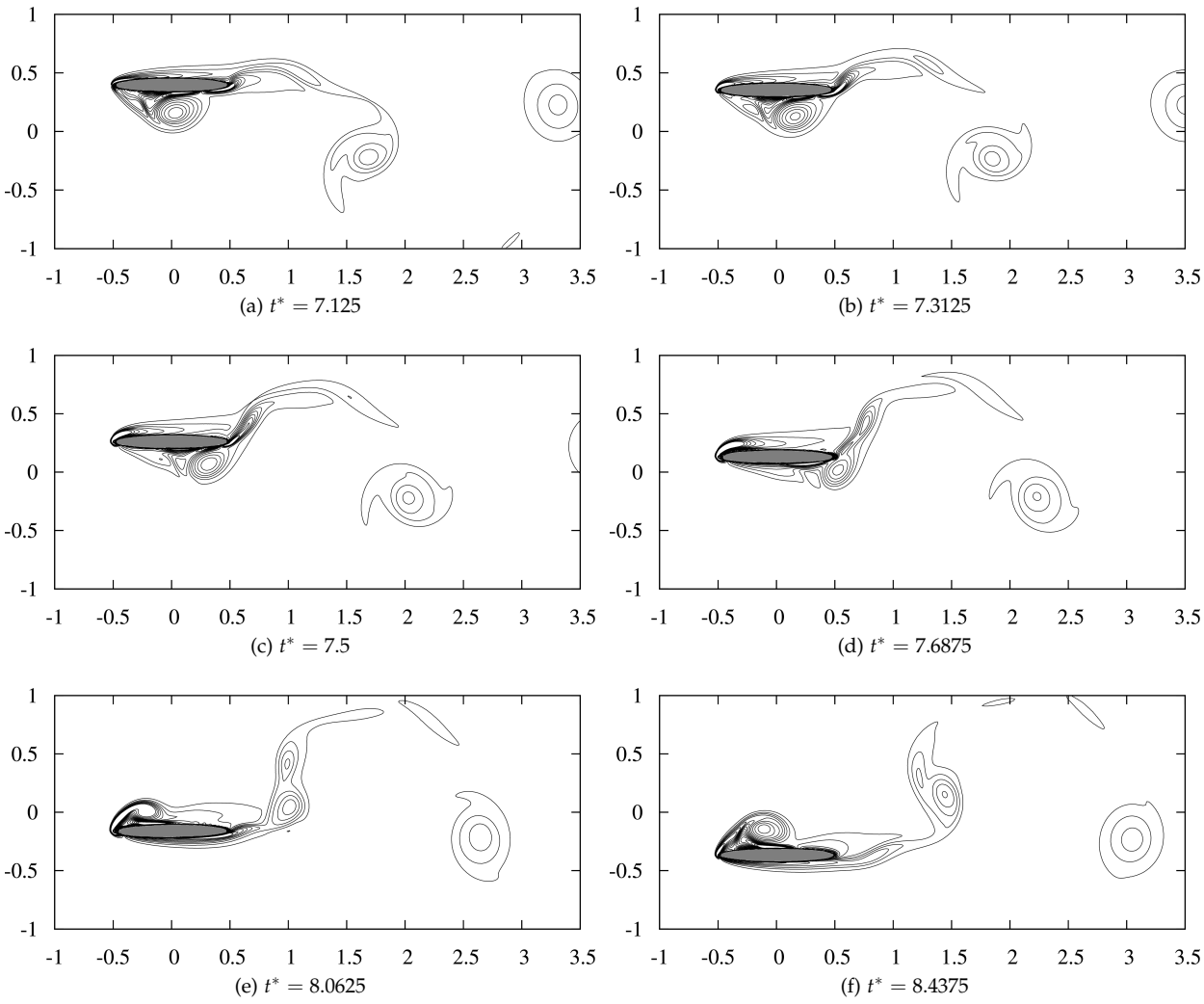


Figure 6: Vorticity field for the downstroke of the heaving airfoil, with contours drawn at levels ± 2, ± 6, ± 10, etc.

Flow due to a flapping airfoil

We consider a flapping airfoil, the motion of which is described by

$$x(t) = \frac{A_0}{2}\cos(2\pi f t)$$
$$\alpha(t) = \alpha_0 + \beta \sin(2\pi f t + \phi),$$

where $x(t)$ is the position of the center of the airfoil and $\alpha(t)$ is the angle made by the airfoil with the line of oscillation. The airfoil is elliptical with a thickness-to-chord ratio of 0.12 and rotates about its center. The Reynolds number is calculated using the maximum translational velocity of the airfoil and the chord length. We consider the case with symmetrical rotation ($\phi = 0$) at Reynolds number 75 and with $A_0/c = 2.8$, $\alpha = \pi/2$, $\beta = \pi/4$ and $f = 0.25$ Hz.

The airfoil has chord length 1 and oscillates at the center of a domain, each side of which is 30 chord-lengths long. The grid is uniform in the region $[-2, 2] \times [-0.52, 0.52]$ with cell width 0.01 and beyond this region it is stretched with a ratio of 1.01 on all sides, resulting in a mesh size of 930×654 cells. The time step used is 0.001.

The vorticity field at different times during the first cycle is shown in Figure **??** and a comparison of the unsteady lift coefficient with the computational and experimental results presented by Wang et al. is plotted in Figure **??**. The experiments were conducted with a three-dimensional wing and both simulations were performed in two dimensions, hence we don't expect them to exactly match. But we note that the computational results follow the expected trend and agree reasonably well with the experimental results.

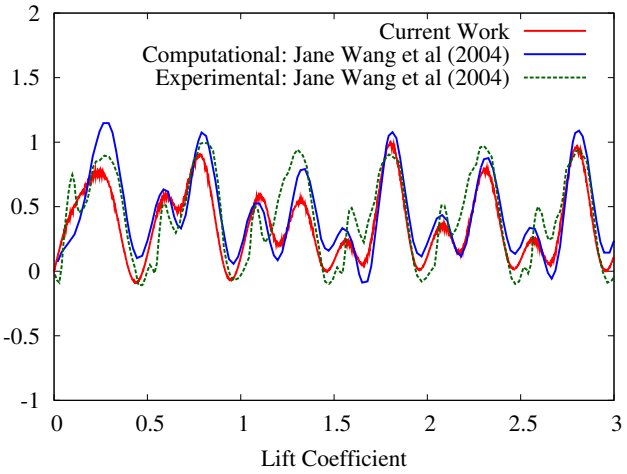


Figure 7: Unsteady lift coefficient for the first three cycles of a flapping airfoil. Time is non-dimensionalized with the period of oscillation and the lift coefficient is calculated by normalizing the lift force with respect to the maximum of the quasi-steady force experienced by the airfoil.

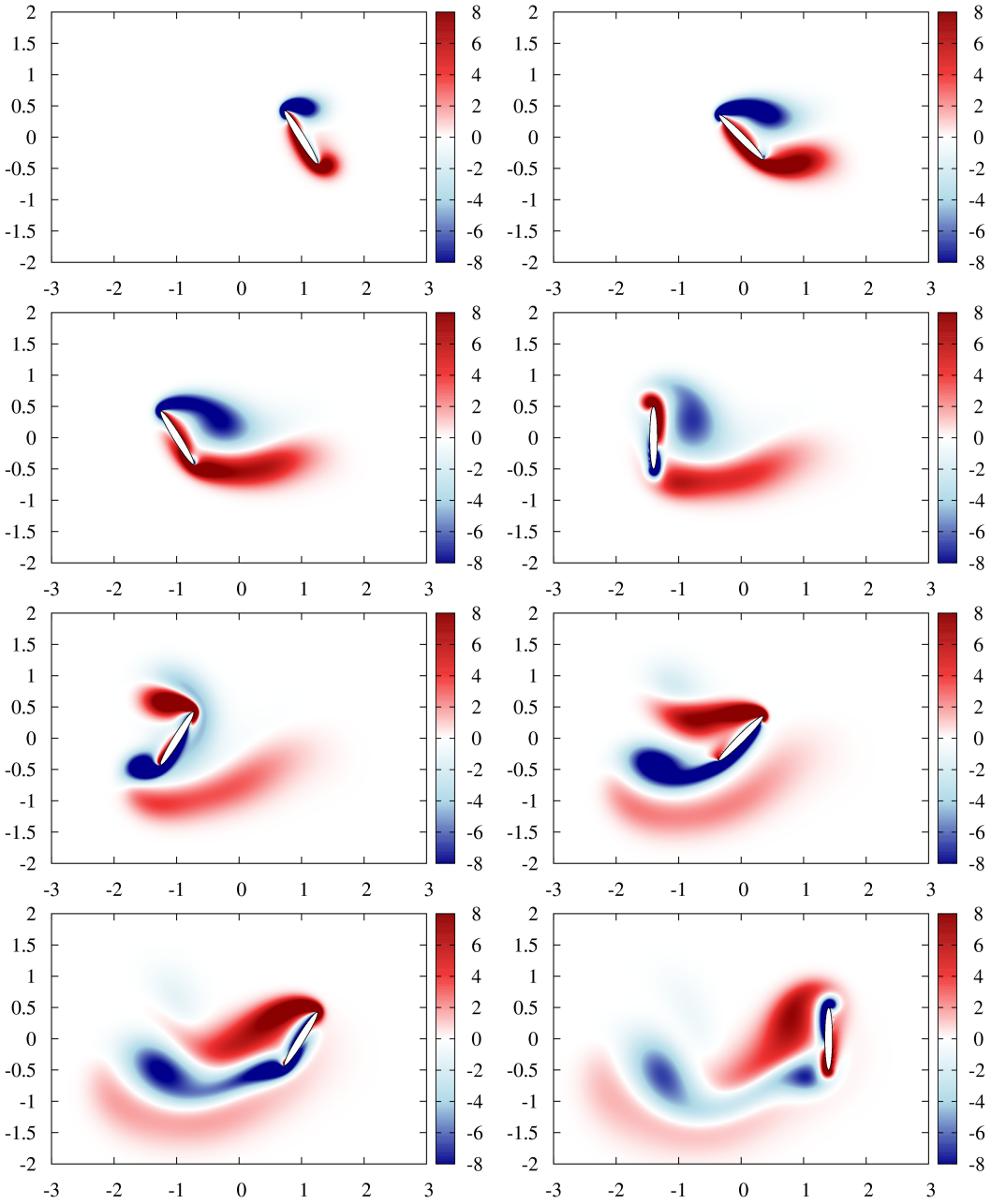


Figure 8: Vorticity field of the flow around a flapping airfoil at equally spaced time instants in the first cycle of flapping: $T = 0.125, 0.25, 0.375, \ldots$.

Implementation details

The various formulations of the IBM that have been published in the open literature can all be written in a form similar to Equation (**??**). We view this form as a general framework to implement a variety of fluid solvers. If there is no immersed body, then the matrices E and E^T are zero and the system solves the unmodified Navier-Stokes equations. In another IBM technique, known as the direct-forcing method,[13] the E matrix is again absent, while the matrix A can be appropriately modified so that the no-slip condition is enforced at the boundary locations that intersect with the grid. In this way, we build the `cuIBM` framework to allow the testing of different solvers and the development of new ones.

[13] E. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comput. Phys.*, 161(1): 35–60, 2000

The code is written in `C++` and uses GPU hardware via `Cuda`. It consists of a base class, `NavierStokesSolver`, which stores all the matrices required for the solution of the Navier-Stokes equations in a rectangular domain without immersed bodies. The class `NSWithBody` inherits from this class and also stores the information regarding any bodies present in the flow domain. The different IBM solvers such as `TairaColoniusSolver` and `FadlunEtAlSolver` inherit from this class. Each derived class makes use of the body information and has its own version of the methods used to generate the sub-matrices A and Q, specific to the solver used.

The flow description and the simulation parameters are supplied to the code using input files. There are four input files:

- Flow file: Specifies the velocity boundary conditions on the domain and the kinematic viscosity of the fluid.
- Domain file: Provides information regarding the grid and flow domain. Currently, only Cartesian grids are supported but the mesh can be stretched exponentially whenever required.
- Body file: Specifies the locations of the boundary points and the rigid-body motion that the body undergoes during the simulation.
- Simulation file: Contains the numerical parameters relevant for the simulation, such as the time step, IBM schemes, time-stepping schemes, etc.

The various time-stepping schemes available are explicit Euler, second-order Adams-Bashforth and third-order Runge-Kutta methods for the convection terms, and explicit Euler, implicit Euler, second-order Adams-Bashforth and Crank-Nicolson schemes for the diffusion terms. All spatial discretizations have been evaluated using central differences.

The matrices and linear systems in cuIBM are stored and solved using the Cusp library,[14] distributed freely by the NVIDIA Corporation under the Apache License. This library provides storage formats and routines for sparse linear algebra and has been optimized to run on NVIDIA GPUs when compiled using the nvcc compiler. The same code can also be compiled to run on CPUs, when it makes use of the Math Kernel Library from Intel. The entire code is written using the Thrust library, now shipping with the Cuda compiler, which consists of a number of basic algorithms that are GPU-optimized versions of those in the C++ Standard Template Library. These functions are templated over the memory type, and the same code can be compiled to run on both CPUs and GPUs.

[14] Nathan Bell and Michael Garland. Cusp: Generic parallel algorithms for sparse matrix and graph computations, 2012. URL http://cusp-library.googlecode.com

We use YAML[15] to parse the human-readable input files. This code is distributed under the MIT license and we have included parts of it in our code base.

[15] Oren Ben-Kiki, Clark Evans, and Brian Ingerson. YAML ain't markup language (YAML) (tm) version 1.2. Technical report, YAML.org, 9 2009. URL http://www.yaml.org/spec/1.2/spec.html

The Cusp library provides various pre-conditioners and solvers, and all sparse linear algebra in cuIBM is performed using Cusp. Several custom Cuda kernels were written so that all operations take place on the GPU and the data transfer between the host and the device is minimized. It is well known that sparse matrix computations are memory-bound, and the larger memory bandwidth provided by GPUs result in a speed-up compared to the code running on a CPU. However, we have not implemented an optimized CPU code and thus are not able to report comparative performance metrics.

We release the cuIBM code under the MIT License, and maintain a Bitbucket repository at https://bitbucket.org/anushk/cuibm/. We also provide MATLAB and Gnuplot postprocessing scripts to plot the various flow variables. User documentation is currently being written.

References

Nathan Bell and Michael Garland. Cusp: Generic parallel algorithms for sparse matrix and graph computations, 2012. URL http://cusp-library.googlecode.com.

Oren Ben-Kiki, Clark Evans, and Brian Ingerson. YAML ain't markup language (YAML) (tm) version 1.2. Technical report, YAML.org, 9 2009. URL http://www.yaml.org/spec/1.2/spec.html.

E. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comput. Phys.*, 161(1):35–60, 2000.

U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comput. Phys.*, 48(3):387–411, 1982. DOI: 10.1016/0021-9991(82)90058-4. URL http://www.sciencedirect.com/science/article/pii/0021999182900584.

P. Koumoutsakos and A. Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *J. Fluid Mech.*, 296:1–38, 1995.

G. C. Lewin and H. Haj-Hariri. Modelling thrust generation of a two-dimensional heaving airfoil in a viscous flow. *J. Fluid Mech.*, 492:339–362, 2003.

R. Mittal and G. Iaccarino. Immersed boundary methods. *Ann. Rev. Fluid Mech.*, 37(1):239–261, 2005. ISSN 0066-4189. DOI: 10.1146/annurev.fluid.37.061903.175743. URL http://arjournals.annualreviews.org/doi/abs/10.1146%2Fannurev.fluid.37.061903.175743.

J. B. Perot. An analysis of the fractional step method. *J. Comput. Phys.*, 108(1):51–58, 1993. ISSN 0021-9991. URL http://www.ecs.umass.edu/mie/faculty/perot/Papers/FSM0.pdf.

C. S. Peskin. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.*, 10(2):252–271, 1972. ISSN 0021-9991. URL http://linkinghub.elsevier.com/retrieve/pii/0021999172900654.

A. M. Roma, C. S. Peskin, and M. J. Berger. An adaptive version of the immersed boundary method. *J. Comput. Phys.*, 153(2):509–534, 1999.

K. Taira and T. Colonius. The immersed boundary method: A projection approach. *J. Comput. Phys.*, 225(2):2118–2137, 2007. ISSN 00219991. DOI: 10.1016/j.jcp.2007.03.005. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999107001234.

D. J. Tritton. Experiments on the flow past a circular cylinder at low Reynolds numbers. *J. Fluid Mech.*, 6(04):547–567, 1959.

Z. J. Wang, J. M. Birch, and M. H. Dickinson. Unsteady forces and flows in low Reynolds number hovering flight: two-dimensional computations vs. robotic wing experiments. *J. Exp. Biol.*, 207(3):449–460, 2004.

C. H. K. Williamson. Vortex dynamics in the cylinder wake. *Ann. Rev. Fluid Mech.*, 28:477–539, 1996.