An implementation of the direct-forcing immersed boundary method using GPU power

Bulent Tutkun and Firat Oguz Edis

Faculty of Aeronautics and Astronautics, Istanbul Technical University, Istanbul, Turkey

ABSTRACT
A graphics processing unit (GPU) is utilized to apply the direct-forcing immersed boundary method. The code running on the GPU is generated with the help of the Compute Unified Device Architecture (CUDA). The first and second spatial derivatives of the incompressible Navier-Stokes equations are discretized by the sixth-order central compact finite-difference schemes. Two flow fields are simulated. The first test case is the simulated flow around a square cylinder, with the results providing good estimations of the wake region mechanics and vortex shedding. The second test case is the simulated flow around a circular cylinder. This case was selected to better understand the effects of sharp corners on the force coefficients. It was observed that the estimation of the force coefficients did not result in any troubles in the case of a circular cylinder. Additionally, the performance values obtained for the calculation time for the solution of the Poisson equation are compared with the values for other CPUs and GPUs from the literature. Consequently, approximately 3× and 20× speedups are achieved in comparison with GPU (using CUSP library) and CPU, respectively. CUSP is an open-source library for sparse linear algebra and graph computations on CUDA.

1. Introduction
The immersed boundary method (IBM) is a numerical method that is currently the focus of many researchers. The IBM is a tool applied to the solution of flow problems with arbitrary boundaries using a Cartesian-style computational grid. That is, the computational grid does not need to conform to the boundaries, with the desired boundary conditions imposed by applying a numerical algorithm in the vicinity of the immersed boundary (Mittal & Iaccarino, 2005). Therefore, this approach effectively avoids the difficulties faced in the grid generation of complex boundaries. Moreover, the computational cost per grid node is generally much lower than that of general purpose, unstructured grid solvers. As a result, because of the offered simplification in the grid generation and other aspects, the CFD community has recently taken an interest in this technique, which was originally aimed at biological flows. With IBM, equations may be solved only on a Cartesian grid, and there is no need to reconstruct the grid in the case of moving or deforming bodies. Although the immersed boundary method was first proposed and implemented by Peskin (1972) to simulate blood flow interacting with the heart, now it is in use for the solution of various problems, such as moving rigid boundaries (Liao, Chang, Lin, & Mcdonough, 2010), flapping airfoils (Yang, He, & Zhang, 2010), heat transfer (Luo, Zhuang, Fan, & Haugen, 2016), viscoelastic fluid (Lee & Wolgemuth, 2016), acoustic wave scattering (Seo & Mittal, 2011), natural convection (Dash & Lee, 2014), and separated turbulent flow (Bernardini, Modesti, & Pirozzoli, 2016). In his method, Peskin used Lagrangian points connected by springs to simulate the elastic boundary. Later, the method was adapted to simulate rigid boundaries by increasing the spring rigidity (Lai & Peskin, 2000) or using a feedback control (Saiki & Biringen, 1996). However, these methods had heavy time-step restrictions due to the stiffness of the system. To avoid this restriction, Mohd-Yusof (1997) first introduced the direct-forcing approach and imposed the velocity boundary condition directly. Afterwards, this method was implemented by many researchers (Balaras, 2004; Iaccarino & Verzicco, 2003; Kim & Choi, 2006;
Tseng & Ferziger, 2003). In direct-forcing IBM, a solid body is immersed in the grid with the use of an added forcing term to the governing equation. The velocity boundary condition on the immersed body is satisfied with the help of the forcing term, which is calculated from the algebraic equations in the discretized problem. An application of a direct-forcing IBM to run on a GPU for the solution of incompressible Navier-Stokes equations is presented in this study.

Spectra of length and time scales of turbulent or highly convective flows are quite wide. Therefore, numerical methods used in the simulation of such flows should be competent to resolve the best part of relevant scales. Grid resolution is the decisive factor for the representation of the length scales and the numerical scheme used defines the accuracy level of the computation. Using finer grids with the standard second-order methods can meet these requirements. However, the efficiency of the scheme may be limited by massive memory demand or numerical error. Low-order numerical methods have difficulties in simulating vortex-dominated flow fields. The main reasons for this are deformation and dissipation of the vortical flow structures as a result of numerical diffusion in the solution algorithm. The application of high-order methods may reduce the grid size and enhance the accuracy. In this respect, compact finite difference methods (Lele, 1992) have become quite popular in computational fluid dynamics (CFD) applications.

Compact finite-difference schemes are generally better in resolving the broad spectrum of length scales in comparison with standard finite-difference schemes of the same order. Thus, compact schemes require a relatively smaller stencil and provide better resolution, especially at higher wave numbers. Therefore, they can combine the robustness of finite-difference schemes and the accuracy of a spectral-like solution in an efficient way. Comprehensive studies focusing on the resolution characteristics of the higher-order compact schemes on a uniform grid have been performed. Compact schemes are applied in the simulation of the various incompressible and compressible flow problems (Boersma, 2011; Hejranfar & Ezzatneshan, 2014; Kalita & Dass, 2011; Meinke, Schroder, Krause, & Rister, 2002; Mishra & Sanyasiraju, 2012; Nagarajan, Lele, & Ferziger, 2003; Wang, Zhang, Ma, Qiu, & Liang, 2016). In compact finite-difference schemes, derivatives are computed implicitly with the utilization not only of function values but also of derivative values at the neighboring nodes. In the present study, the sixth-order central compact finite difference schemes are used for the solutions of incompressible flow equations.

The increase in the computational power of computer systems is one of the most prominent factors for the successful applications of large-scale CFD problems. The emergence of graphical processing units (GPU) as a scientific computing platform in recent years is a good example of such advancements. At the beginning, they provided service with continuously increasing gaming-content performance to gamers in various simulations. The continuous increase in the computational power of GPUs has widened their use from gaming to applications with heavy computational burdens in the field of computational sciences. GPUs, with their many core structures and outstanding processing performance, offer a considerable alternative for scientific computation. It is well known that GPUs are originally designed to achieve high performance at processing big graphics data sets. Since GPUs inherently have an ability to compute in parallel, they can provide fast and low-cost solutions especially for large-scale computational problems. Numerous applications from various engineering fields that exploit the parallel performance of GPUs can be found in the literature (Owens et al., 2007). In the field of CFD, some promising implementations are also realized with various flow-field simulations. One example is in Hagen, Lie, and Natvig (2006), which represents one of the early applications in compressible fluid flows. Brandvik and Pullan (2007, 2008), using GPU computing, performed 2D and 3D simulations of Euler equations and made performance comparisons between GPUs and CPUs. A hypersonic-flow simulation on a GPU platform may be found in the research article of Elsen, Legresley, and Darve (2008). They used an NVIDIA 8800GTX GPU and simulated a hypersonic vehicle in cruise at Mach 5 to demonstrate the capabilities of GPU code. They achieved an acceleration of 20x for the Euler equations. Additionally, several recent CFD applications can be seen in Wang, Abel, and Kaehler (2010); Xian and Takayuki (2011); Karantasis, Polychronopoulos, and Ekaterinaris (2014); Ma, Wang, and Pu (2014); Liu, Zhong, and Xu (2016), and Jude and Baeder (2016).

In the present study, simulations of two flow fields around a square and a circular cylinder are realized using the hardware and numerical tools mentioned. The flow past a square cylinder is chosen as a test case in order to see behaviors of the high-order compact scheme and the immersed boundary method under the influence of sharp corners. As expected, the high-order compact schemes bring about spurious oscillations in the neighborhood of the immersed boundary. This issue is alleviated by using mirror like and decaying velocity conditions for the solid side of the immersed body to create an internal flow field that ensures no-slip condition on the immersed body. From the point of the computational performance of the GPU, there are two important complications. The first one is the coalescing issue
regarding the memory of the GPU. To overcome this issue, an approach making use of the shared memory recommended in Tutkun and Edis (2012) is applied. Second, since the Poisson equation solver is the most time-consuming part of the code, performance of this solver is another issue which affects the overall performance of the solution procedure. Hence, a special sparse matrix storage format (ELLPACK) is incorporated into the code, thereupon developed Poisson solver achieves the higher performance by means of this storage format.

Second test case, the flow past a circular cylinder, is chosen to figure out the effects of sharp corners on the accuracy of the simulation. The flow around a circular cylinder is simulated with the same conditions and processes as in the square cylinder case in order to be able to observe the behavior differences of the numerical method under the effects of sharp-cornered and smooth-contoured bodies.

These simulations show the accuracy limits of the direct-forcing IBM and the performance improvements provided by the GPU computing environment. The GPU used in the computations is a Tesla C1060 produced by NVIDIA for general-purpose computing.

2. Governing equations

In the context of this study, the continuity and Navier-Stokes equations for unsteady, incompressible flows are considered to be:

\[ \nabla \cdot \mathbf{V} = 0 \]  
(1)

\[ \frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V} = -\frac{1}{\rho} \nabla p + \partial \nabla^2 \mathbf{V} + \mathbf{F} \]  
(2)

where \( \mathbf{V} \) is the fluid velocity, \( \rho \) is the density, \( p \) is the pressure, \( \partial \) is the kinematic viscosity and \( \mathbf{F} \) is the forcing term obtained from the immersed boundary method. The equations are spatially discretized by using sixth-order, compact finite-difference schemes. In addition, the second-order Adams-Bashforth method with a fractional step approach is applied for the time integration. The details of these discretizations are given in the subsequent sections.

3. Numerical methods

3.1. Spatial discretization

A high-order, central, compact finite-difference scheme is applied for spatial discretization on a non-staggered grid system. Compact finite-difference schemes provide more accuracy than the standard finite-difference schemes. To achieve this, compact schemes mimic the behavior of spectral methods that couple derivative terms over all grid nodes. In that way, the compact scheme couples the derivatives of the neighboring nodes to reach an improved accuracy. Lele (1992) introduced the compact finite-difference schemes and investigated their accuracy features. Starting from the Hermitian formula (Hoffmann & Chiang, 2000) and using the Taylor series expansions for the terms, we can come up with a general, five-point formulation for the approximation of the first derivative:

\[ \beta f''_{i-2} + \alpha f''_{i-1} + f'_i + \alpha f''_{i+1} + \beta f''_{i+2} = \epsilon \frac{f_{i+3} - f_{i-3}}{6h} + b \frac{f_{i+2} - f_{i-2}}{2h} + a \frac{f_{i+1} - f_{i-1}}{2h} \]  
(3)

where \( f \) and \( f' \) show a flow variable and its derivative, respectively. \( h \) is the step size in the corresponding direction. \( \alpha \) and \( \beta \) define the characteristic of generated system and it may be either a tri-diagonal or a penta-diagonal one. If the periodic boundary conditions are implemented, then the scheme produces a cyclic tri-diagonal or a cyclic penta-diagonal system. Provided that the coefficients are chosen as in the following expression, then the sixth-order compact formulation is obtained:

\[ \frac{1}{9} f'_{i-1} + f'_i + \frac{1}{9} f'_{i+1} = \frac{1}{9} \left( \frac{f_{i+2} - f_{i-2}}{4h} \right) + \frac{14}{9} \left( \frac{f_{i+1} - f_{i-1}}{2h} \right) \]  
(4)

Compact schemes for the second derivative can be formed in a similar way. Again, we can start with an expression including the second derivatives and function values of the neighboring points. This relation form is also similar to the one for the first derivatives:

\[ \beta f''''_{i-2} + \alpha f''''_{i-1} + f''''_i + \alpha f''''_{i+1} + \beta f''''_{i+2} = \epsilon \frac{f_{i+3} - 2f_i + f_{i-3}}{9h^2} + b \frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} + a \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \]  
(5)

Again, the sixth-order scheme is shown below:

\[ \frac{2}{11} f''''_{i-1} + f''''_i + \frac{2}{11} f''''_{i+1} = \frac{3}{11} \left( \frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} \right) + \frac{12}{11} \left( \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \right) \]  
(6)

For non-periodic boundaries, one-sided schemes are applied on the boundary because the stencil of the inner compact scheme extends out of the boundary. In the
In this study, the direct-forcing IBM is utilized in the... boundary scheme is preserved by utilizing these schemes. As a result, the third-order, one-sided, compact scheme and the fourth-order, central, compact scheme are applied for the boundary point \( i = 1 \), and the near-boundary point \( i = 2 \), respectively. These schemes are presented below for the first and second derivatives:

1st derivative, \( i = 1 \) (3rd order)
\[ f'_1 + 2f'_2 = \frac{1}{2h}(-5f_1 + 4f_2 + f_3) \]  
\[ (7) \]

1st derivative, \( i = 2 \) (4th order)
\[ \frac{1}{4}f'_1 + f'_2 + \frac{1}{4}f'_3 = \frac{3}{4h}(f_2 - f_1) \]  
\[ (8) \]

2nd derivative, \( i = 1 \) (3rd order)
\[ f''_1 + 11f''_2 = \frac{1}{h^2}(13f_1 - 27f_2 + 15f_3 - f_4) \]  
\[ (9) \]

2nd derivative, \( i = 2 \) (4th order)
\[ \frac{1}{10}f''_1 + f''_2 + \frac{1}{10}f''_3 = \frac{6}{5h^2}(f_2 - 2f_3 + f_4) \]  
\[ (10) \]

Similar formulations are implemented for the right boundary at \( i = N \) and the near-boundary point at \( i = N - 1 \).

### 3.2. Temporal discretization and the immersed boundary method

In this study, the direct-forcing IBM is utilized in the solution of incompressible Navier-Stokes equations. The essence of direct-forcing is the imposition of the boundary velocity \( V_b \) on the solution of the velocity for the solid boundary. This imposition is applied via a force term that is added to the governing equations. The x-component of the incompressible Navier-Stokes equations is:

\[
\frac{\partial u}{\partial t} = -\left( u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) - \frac{1}{\rho} \frac{\partial p}{\partial x} + \vartheta \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + F_x
\]

where \( F_x \) is the forcing term for the IBM and is inserted into the equation to ensure the proper boundary velocity value on the surface of the solid body. If we simply discretize this equation in time:

\[
\frac{u^{n+1} - u^n}{\Delta t} = RHS + F_x
\]

RHS in this equation includes the convection, diffusion and pressure terms. From this relation, the forcing value ensuring the velocity boundary condition on the wall can be easily written as:

\[
F_x = \varepsilon \left( -RHS + \frac{V_b^{n+1} - V^n}{\Delta t} \right)
\]

In this expression, \( \varepsilon = 1 \) in the solid-body region, and \( \varepsilon = 0 \) everywhere else (Laizet & Lamballais, 2009). Therefore, with the use of the forcing term \( F_x \), the velocity boundary condition can be imposed directly onto the solid surface. As a result, the boundary conditions on the wall are satisfied at every time step. This is the general overview of the IBM used in this study. Computational structure of the GPU performs much more efficiently with an explicit time integration method. The first and a popular alternative is the Runge-Kutta methods. However, Runge-Kutta methods that necessitate the solution of the Poisson equation more than once per one time-step have fatal effects on the performance of the solver. Therefore, time integration is realized with the use of a second-order Adams-Bashforth method with a fractional step approach. We can combine the time integration with the governing equations to explain the IBM as used in this study.

The \( F \) in (2) is the direct forcing vector that is described before. With the use of this forcing term, the required boundary conditions on the solid surfaces are directly imposed. If both the convection and the diffusion terms are represented by \( A \):

\[
A = -(V \cdot \nabla V) + \vartheta \nabla^2 V
\]

The time advancement of the momentum equation in (2) may be given as:

\[
\frac{V^* - V^n}{\Delta t} = \frac{3}{2} A^n - \frac{1}{2} A^{n-1} - \frac{1}{\rho} \nabla p^n + F^n
\]

\[
\frac{V^{**} - V^*}{\Delta t} = \frac{1}{\rho} \nabla p^n
\]

\[
\frac{V^{n+1} - V^{**}}{\Delta t} = -\frac{1}{\rho} \nabla p^{n+1}
\]

In this approach, the expression for the direct forcing term related to IBM may be given by:

\[
F^n = \varepsilon \left( -\frac{3}{2} A^n + \frac{1}{2} A^{n-1} + \frac{1}{\rho} \nabla p^n + \frac{V_b^{n+1} - V_n}{\Delta t} \right)
\]

\[\text{(18)}\]
As mentioned before, $\varepsilon$ is defined as 1 in the solid-body region and 0 everywhere else. With the insertion of this term into (15), the desired velocity boundary condition in the region prescribed by $\varepsilon$ is satisfied in the first step of the fractional step method. Normally, in a standard fractional-step method, the incompressibility condition $\nabla \cdot \mathbf{V}^n + \frac{1}{\Delta t} = 0$ may be satisfied with the solution of a Poisson equation in the form of:

$$\nabla \cdot \nabla p^n = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{V}^{*\ast}$$  \hspace{1cm} (19)

When IBM is incorporated into the computational method, this equation is transformed into:

$$\nabla \cdot \nabla p^{n+1} = \frac{\rho}{\Delta t} \nabla \cdot [(1 - \varepsilon) \mathbf{V}^{*\ast}]$$  \hspace{1cm} (20)

When $\varepsilon = 0$, the standard Poisson equation is obtained. However, in the solid-body region, $\varepsilon = 1$ (20) gives the Laplace equation. For the discretization of Equation (20), a second-order, centered, finite-difference scheme is utilized. Solution of the discretized equation is realized using a classical conjugate-gradient algorithm on GPU.

The use of the high-order, centered scheme with the direct-forcing approach may amplify the spurious oscillations occurring in the vicinity of the wall on which the boundary condition is imposed by the use of forcing terms. Therefore, an internal flow field supporting a no-slip boundary imposition on the immersed boundary is created to mitigate this issue. By using this approach, first proposed by Parnaudeau, Lamballais, Heitz, and Silvestrini (2004), we can create an internal flow similar to that of the mirror condition.

Therefore, the target-velocity values of the nodes inside the immersed boundary are specified with the help of the external flow field. This treatment may be seen in Figure 1 for the square and circular cylinder cases. Here, ‘m’ is a function used to cancel out a possible singularity at the centers of the cylinders. Surely, some other forms of the function ‘m’ may be used provided that the three following conditions are satisfied: (a) $m(1/2) = 1$; reverse condition in the vicinity of the body surface, (b) $m(0) = 0$; singularity cancellation and (c) $0 \leq m(r) \leq 1$ with moderate first and second derivatives for $0 \leq r \leq \frac{1}{2}$. In the square cylinder case, because the immersed boundaries and the grid lines coincide, specifying the internal flow is performed along each grid line. However, in the circular cylinder case, a polar coordinate system is used to determine the internal flow field. In addition to this, we apply zero gradient condition for the pressure. This is performed by conditioning first two grid nodes inside the immersed boundary with the pressure values coming from the corresponding nodes outside the body.

To assess the success of this approach, a seventh-order polynomial function is used to see the error which compact scheme (Equation 4) together with a mirror-like boundary condition generates. The seventh-order polynomial is in the form of:

$$f(x) = \frac{x^7}{7} - \frac{x^6}{2} + \frac{17x^5}{25} - \frac{9x^4}{20} + \frac{274x^3}{1875} - \frac{12x^2}{625}$$  \hspace{1cm} (21)

over the domain $x = [0, 1]$. Two ghost nodes are identified at the boundaries of the domain in order to be able to apply the Equ. (4) over the whole interval. Also this mirror-like condition mimics the treatments for the immersed boundary in the vicinity of the solid wall. Mentioned boundary condition for the left boundary ($i = 1$) may be given as:

$$f_0 \rightarrow -f_2, f_{-1} \rightarrow -f_3, f'_0 \rightarrow f'_2$$

This shows the relations just for the left boundary, corresponding relations for the right boundary can be identified simply. The maximum error occurs at the boundaries as expected because of the applied ghost node conditions.

---

**Figure 1.** Handling of the nodes within the immersed boundary for the square and circular cylinders.
Therefore reduction in the error value \( (f_{\text{exact}} - f_{\text{computed}}) \) for the left boundary point is presented in Figure 2 (left). Spoiling effects of the implemented boundary conditions reduce sixth-order accuracy of the compact scheme to nearly second-order accuracy. Although we face a degradation in accuracy of the compact scheme, a sixth-order accuracy can be recovered outside from the border vicinity as shown in Figure 2 (right).

4. GPU and code design

Computations in this study are done on a NVIDIA Tesla C1060 GPU. This device has 30 multi-processors, and each of them comprises eight scalar processors, clocked at 1.3 GHz. Our host system accommodates an AMD Phenom X4 9850 CPU and 8 GB memory. The code for the simulations of flow problems is generated by using CUDA toolkit that is developed by the NVIDIA in order to make programming the GPUs straightforward. The GPU with the capability of performing large-scale computations in parallel is considered as a coprocessor designed to supplement the abilities of the main CPU. In this approach, compute-intensive parts of the code running on the CPU are directed to the GPU by employing a kernel (may be viewed as a function) executed on the GPU by many-core computational structure.

In the present study, structure of the threads generated on the GPU is similar to that applied in Tutkun and Edis (2012). One thread is assigned to each finite-difference node in the computational domain. As a result of this approach, all calculations required for a node are done by the same thread. The CUBLAS library for CUDA is utilized for the solutions of the cyclic-tridiagonal and tridiagonal systems resulting from the high-order, compact schemes. In order to solve the Poisson equation, a conjugate-gradient algorithm is implemented through a custom kernel in which the ELLPACK format (Bell & Garland, 2009) is also used to store sparse matrices. The ELLPACK format provides an increase in performance as will be shown in the corresponding section. Double precision is applied in the conjugate-gradient algorithm to ensure convergence. Moreover, the shared memory with its on-chip nature and increased speed compared to the global memory is exploited to alleviate certain non-coalescing issues.

5. Applications and performance comparisons

Fluid flow around bluff bodies is one of the most commonly encountered flow problems in various engineering fields. It has many practical applications in terms of CFD. One of the main characteristics of this configuration is the flow separation at the immersed body surface. Thus, in turn, it leads to a remarkable wake region that is characterized mainly by the shedding vortices from the upper and lower sides of the body. Depending on the flow system, this wake region may present certain benefits, such as enhanced heat and mass transfer, but at the same time, it may cause certain complicating issues, such as unwanted vibrations. Therefore, predicting the wake region accurately is quite important. In this study, the flows around two important bluff bodies, a square and a circular cylinder, are simulated using the immersed boundary method.

5.1. Flow around a square cylinder

Simulation of the flow around a square cylinder is applied to evaluate the immersed boundary method on GPU for \( \text{Re} = 100 \). Sixth-order, compact, central finite-difference schemes are utilized for both the first and the second spatial derivatives for the discretization of the incompressible Navier-Stokes equations. The flow domain of this test case and the applied boundary conditions are given in Figure 3. The square cylinder is placed between
Figure 3. Flow domain and boundary conditions for the square cylinder test case.

Table 1. Grid convergence test for the square cylinder test case (Re = 100).

| Grid size   | Cl_{rms} | Cl_{ave} | St.  |
|-------------|----------|----------|------|
| 192 x 96    | 0.21     | 1.35     | 0.134|
| 256 x 128   | 0.24     | 1.40     | 0.140|
| 384 x 192   | 0.26     | 1.41     | 0.146|
| 512 x 256   | 0.26     | 1.41     | 0.146|

The values of the dimensions in the streamwise and the stream-normal directions are X/D = 25.5 and Y/D = 12.7 respectively.

At the inlet (i = 1), the applied boundary conditions are \( u = U_{\infty} \), \( v = 0 \) and \( \partial p / \partial x = 0 \). At the outlet (i = N), the boundary conditions are \( \partial u / \partial x = \partial v / \partial x = 0 \) and \( p = 0 \). Moreover, periodic boundary conditions are imposed on the remaining boundaries. Uniform grids are used for the computations of this test case. A grid convergence test is also performed, as shown in Table 1 for the Re = 100 case. The Strouhal number (St) in Table 1 is also an important non-dimensional parameter giving information about the vortex shedding frequency of the flow, and it is defined as:

\[
St = \frac{fD}{U_{\infty}}
\]  

where, \( f \), \( D \) and \( U_{\infty} \) stand for the shedding frequency, the cylinder dimension and the free-stream velocity, respectively.

The Reynolds number is defined as \( Re = U_{\infty}D/\vartheta \). As a result of this study, the total node numbers along the Cartesian coordinates (x, y) are selected as 384 x 192, respectively. The near-wall part of the computational grid created for the square cylinder test case may be seen in Figure 4. In vortex-dominated flow fields, it is very important to accurately calculate not only the region near the body but also the wake region. This is also another factor for the usage of the high-order, compact scheme that has high accuracy and low numerical dissipation in the region far from the immersed boundary. In this respect, using the high-order compact scheme also provides an opportunity to apply the uniform grid throughout the flow domain.

In Table 2, the \( \Delta x \) values showing the distance of the first node to the solid body are indicated for various studies in the literature and the present study. The table also provides the numerical methods used for the studies mentioned. As shown in the table, the distance of the first node to the body is at least six times higher in this study than in the others.

In the context of this study, Figure 5 shows the instantaneous streamlines of the flow domain for Re = 100. The form of the antisymmetric wake flow, called the von
Table 2. First node distance from the body for the square cylinder test case.

| Studies                           | Δx  | Method                                      |
|-----------------------------------|-----|---------------------------------------------|
| Sharma and Eswaran (2004)         | 0.01| Finite volume method. 2nd order in space.   |
| Sohankar, Norberg, and Davidson (1998) | 0.004| Finite volume method. 2nd–3rd order in space. |
| Sahu, Chhabra, and Eswaran (2009) | 0.01| Finite volume method. 2nd order in space.   |
| Singh, De, Carpenter, Eswaran, and Muralidhar (2009) | 0.007| Finite volume method. 2nd order in space.   |
| Sen, Mittal, and Biswas (2011)    | 0.001| Finite element method.                      |
| Present                           | 0.066|                                             |

Figure 5. Instantaneous streamlines for the square cylinder test case.

Karman vortex street, is presented. As shown in Figure 5, with the purpose of describing the characteristics of the immersed boundary method, this illustration does not involve a square cylinder in the flow domain as an object, but the flow behaves as if there is one.

The flow around a square cylinder shows similar characteristics with the flow around a circular cylinder if we consider instabilities only. However, the separation process and dependence of the shedding frequency on the Reynolds number differs greatly. Unlike the circular cylinder, the separation points of the square cylinder are fixed at its leading or trailing edges. Moreover, the wake width just at the back side of cylinder is at least one diameter, but for the circular cylinder it is less than half a diameter. As a result, the vortex formation region is much longer and broader for the square cylinder than for the circular cylinder. A totally attached fluid flow is observed over the square cylinder at Re = 1. However, at around Re ≈ 2 – 3 the flow starts separating at the trailing edge of the cylinder. Consequently, a confined recirculating region consisting of two symmetric vortices develops behind the cylinder. Size of the recirculating region increases with Reynolds number and the flow shows steady characteristics. However, at a critical onset Reynolds number the two vortex structure begins to exhibit an unstable behavior and the wake starts oscillating periodically. This periodic phenomenon is known as vortex shedding. In Sohankar, Norberg, and Davidson (1997), for square cylinder flow the reported experimental value for the onset of vortex shedding is Re = 47 ± 2.

Moreover, Figure 6 (a)–(h) depicts the instantaneous streamlines in the vicinity of a square cylinder for a full vortex shedding cycle. In Figure 6, there are also the corresponding images below each phase of the cycle, showing the results of another computational study (Sharma & Eswaran, 2004). With its eight consecutive frames, Figure 6 covers a vortex shedding period. The centers and saddles related to the vortex shedding are also indicated in the figure. In addition, the process of instantaneous ‘alleyways’ mentioned in Perry, Chong, and Lim (1982) is also demonstrated in the figure. In this process, the fluid
is pulled into the recirculation region from both the top and the bottom of the cylinder, successively, through the alleyways.

As an example, two of these alleyways are shown by the red arrows in Figure 6 (b) and (e). For instance, looking at the figure, it can be seen that the period begins with a vortex maturing and shedding from the lower side of the cylinder. At the same time, during this interval, the fluid is constantly pulled into the recirculation region from the upper side. After the vortex sheds, the same process starts again but with the opposite sides. This time, the vortex matures and sheds from the upper side of the cylinder, and the fluid is pulled into the recirculation region from the lower side.

In addition, Table 3 shows a comparison of the recirculation length, defined as the streamwise distance from the base of the square cylinder to the re-attachment point along the wake centerline. Recirculation length regarding Table 3 is based on the time-averaged flow field since the flow is unsteady. The table shows that the obtained recirculation length is in good agreement with the results of other computational studies.

In this context, Table 4 shows a comparison made between various previous studies and the present one in terms of Strouhal numbers for various Reynolds numbers. As the table shows, the values obtained for St are consistent with those from other studies.

Furthermore, because the no-slip boundary condition is not enforced strictly on the body during the solution process, this condition is satisfied approximately. One reason for this is the use of the high-order, compact, central finite-difference scheme with the direct-forcing approach. Although it is highly accurate in space, it may not be inherently as good at resolving the discontinuities because of its quasi-spectral behavior. Internal treatment of the immersed boundary applied here may smooth the discontinuity within limits. However, in the case of the square cylinder, the sharp corners of the cylinder have disturbing effects on the treatment. Consequently, the penetration of certain streamlines into the immersed boundary may occur, as shown in Figure 6. As a result of this, although the wake region and the vortex shedding are simulated accurately, this issue somewhat deteriorates the force calculations. The RMS of the lift coefficients and the time-averaged drag coefficients may be seen in Table 5, along with the results from other studies.

In the present study, forces on the immersed boundary are calculated by a surface force integration approach. Immersed boundary is discretized into elements with sizes same with the grid spacing. Then, by using these nodal values, we calculate velocity derivatives at the surface through one sided-differencing. So in this respect the surface force distribution is evaluated directly. Since we are doing this calculation after the whole flow field

| Study                              | Recirculation length (L_r) |
|------------------------------------|-----------------------------|
| Yoon, Yang, and Choi (2010)        | 1.92                        |
| Sharma and Eswaran (2004)          | 1.92                        |
| Robichaux, Balachandar, and Vanka (1999) | 1.94                             |
| Present                            | 1.91                        |

| Study                              | St (Re = 100) | St (Re = 150) |
|------------------------------------|---------------|---------------|
| Sharma and Eswaran (2004)          | 0.148         | 0.160         |
| Sohankar et al. (1998)             | 0.146         | 0.159         |
| Sahu et al. (2009)                 | 0.147         | 0.156         |
| Singh et al. (2009)                | 0.145         | 0.160         |
| Sen et al. (2011)                  | 0.146         | 0.161         |
| Robichaux et al. (1999)            | 0.146         | 0.161         |
| Franke, Rodi, and Schonung (1990)  |               |               |
| Present                            | 0.146         | 0.161         |

![Figure 7. Power spectrum of C_l (left) and C_d (right) for the square cylinder test case.](image)
Table 5. Comparison of $C_{l_{rms}}$ and $C_{d_{ave}}$ for the square cylinder test case.

| Study                              | $Re = 100$ | $Re = 150$ |
|------------------------------------|------------|------------|
| Sharma and Eswaran (2004)          | 0.19       | 0.29       |
| Sohankar et al. (1998)             | 0.23       | 0.23       |
| Sahu et al. (2009)                 | 0.19       | 0.29       |
| Singh et al. (2009)                | 0.16       | 0.28       |
| Sen et al. (2011)                  | 0.19       | 0.30       |
| Saha, Biswas, and Muralidhar (2003)| 0.12       | 0.27       |
| Present                            | 0.21       | 0.33       |

has been solved, it is actually a post-processing step. It is obvious that the results from this approach depend on the resolution of the surface discretization and the position of the elements and nodes. The dimensionless lift and drag coefficients are defined by:

$$
C_l = \frac{F_l}{0.5\rho U_\infty^2 D} \quad \text{and} \quad C_d = \frac{F_d}{0.5\rho U_\infty^2 D}
$$

(23)

Here $F_l$ and $F_d$ are the lift and the drag forces respectively. The table also shows that the $C_{l_{rms}}$ value deviates much more than the $C_{d_{ave}}$ value.

5.2. Flow around a circular cylinder

As mentioned in the previous section, the force calculations of the square cylinder case show discrepancies in regard to the values from other studies in the literature. Therefore, to observe the possible effects of the sharp corners of the square cylinder, in this section we simulate the flow around a circular cylinder to obtain the Strouhal numbers and force coefficients. The flow domain and the applied boundary conditions are the same as in the square-cylinder case. Near-wall part of the computational grid created for the circular cylinder test case may be seen in Figure 8.

Figure 9 shows the instantaneous streamlines and the $u$ velocity contours of the flow domain for $Re = 100$. The form of the antisymmetric wake flow, called the von Karman vortex street, can be seen. The internal treatment approach used for the nodes in the vicinity of the immersed boundary provides a better representation of the boundary in the circular-cylinder case; correspondingly, the calculated Strouhal numbers and the force coefficients show good agreement with other studies. Related comparisons may be seen in Tables 6 and 7 for the Strouhal numbers and the force coefficients, respectively. We can say that the smooth contour of the circular cylinder is more appropriate for the direct-forcing immersed boundary method in which high-order, central, compact finite-difference schemes are implemented for the spatial discretization.

5.3. Computational breakdown and performance

Generally, the solution process of the Poisson equation takes the vast majority of the running time for one time step. Therefore, the performance of the solution process of this equation directly affects the overall performance of the time-dependent incompressible Navier-Stokes equation solver. Table 8 shows the timing breakdown of one time step in the solution of the incompressible Navier-Stokes equations. It can be seen that the Poisson equation solution is the most decisive part in terms of the general performance. This points out the importance of using efficient linear-algebra algorithms in the solution
Table 6. Comparison of the Strouhal numbers for the circular cylinder test case.

| Study | St (Re = 100) | St (Re = 150) |
|-------|---------------|---------------|
| Liu, Zheng, and Sung (1998) | 0.164 | 0.184 |
| Ren, Wu, Shu, and Yang (2012) | 0.164 | 0.192 |
| Braza, Chassaing, and Minh (1986) | 0.164 | 0.192 |
| Lai and Peskin (2000) | 0.165 | 0.195 |
| Zhang and Zheng (2007) | 0.195 | 0.193 |
| Present | 0.165 | 0.193 |

Table 7. Comparison of Cl and Cd for the circular cylinder test case.

| Study | Re = 100 | Re = 150 |
|-------|----------|----------|
| Cl | Cd | Cl | Cd |
| Liu et al. (1998) | ±0.34 | 1.35 | ±0.59 | 1.44 |
| Ren et al. (2012) | ±0.35 | 1.34 | ±0.59 | 1.44 |
| Braza et al. (1986) | ±0.28 | 1.33 | ±0.59 | 1.44 |
| Lai and Peskin (2000) | ±0.303 | 1.35 | ±0.59 | 1.44 |
| Chiu, Lin, and Sheu (2010) | 1.37 | 1.39 |
| Silva, Silveira-Neto, and Damasceno (2003) | 1.37 | 1.39 |
| Su, Lai, and Lin (2007) | 1.37 | 1.39 |
| Present | ±0.35 | 1.37 | ±0.60 | 1.38 |

Table 8. Timing breakdown of one time step for the square cylinder test case.

| % of Computation Time |
|------------------------|
| Poisson solution           | 81.0% |
| Solution of linear systems | 7.9%  |
| Forming right sides       | 4.4%  |
| Calculating solution vectors | 3.8% |
| Other parts               | 2.9%  |

Further analysis of the timing breakdown of the relevant parts of the code yields Table 9. This table, similar to the previous one, also indicates a timing breakdown of a certain part in the code in which the conjugate-gradient iterative solver occurs. It may also be noted from the table that sparse matrix-vector multiplication is the most time-consuming section of the iterative process (≈ 74%). The ELLPACK sparse matrix storage format is applied in the custom kernel generated for the process of sparse matrix-vector multiplication. The ELLPACK format is more efficient than other commonly used formats (COO, CSR, etc.) for the GPU computations. In this format, an important factor is the maximum number of nonzero elements in a row (say K). As a result, non-zero elements of a sparse matrix M by N are stored into an M x K data array. Rows that do not have K nonzero elements are padded with zeroes. Likewise, column indices are held in another array with a similar padding approach. In addition to this, if the data storing is performed in the form of column-major order, then nearly full memory coalescing is provided on the GPU, which in turn produces higher performance. Moreover, another vector containing the exact number of nonzero elements of each row is also generated. Therefore, unnecessary operations of padded zeroes are prevented with the incorporation of this additional vector into the algorithm. This format is represented in Figure 10. This figure shows the arrays of data, column indices, and nonzero elements per row for a general sparse matrix A.

Further analysis of the timing breakdown of the relevant parts of the code yields Table 9. This table, similar to the previous one, also indicates a timing breakdown of a certain part in the code in which the conjugate-gradient iterative solver occurs. It may also be noted from the table that sparse matrix-vector multiplication is the most time-consuming section of the iterative process (≈ 74%). The ELLPACK sparse matrix storage format is applied in the custom kernel generated for the process of sparse matrix-vector multiplication. The ELLPACK format is more efficient than other commonly used formats (COO, CSR, etc.) for the GPU computations. In this format, an important factor is the maximum number of nonzero elements in a row (say K). As a result, non-zero elements of a sparse matrix M by N are stored into an M x K data array. Rows that do not have K nonzero elements are padded with zeroes. Likewise, column indices are held in another array with a similar padding approach. In addition to this, if the data storing is performed in the form of column-major order, then nearly full memory coalescing is provided on the GPU, which in turn produces higher performance. Moreover, another vector containing the exact number of nonzero elements of each row is also generated. Therefore, unnecessary operations of padded zeroes are prevented with the incorporation of this additional vector into the algorithm. This format is represented in Figure 10. This figure shows the arrays of data, column indices, and nonzero elements per row for a general sparse matrix A.

Further analysis of the timing breakdown of the relevant parts of the code yields Table 9. This table, similar to the previous one, also indicates a timing breakdown of a certain part in the code in which the conjugate-gradient iterative solver occurs. It may also be noted from the table that sparse matrix-vector multiplication is the most time-consuming section of the iterative process (≈ 74%). The ELLPACK sparse matrix storage format is applied in the custom kernel generated for the process of sparse matrix-vector multiplication. The ELLPACK format is more efficient than other commonly used formats (COO, CSR, etc.) for the GPU computations. In this format, an important factor is the maximum number of nonzero elements in a row (say K). As a result, non-zero elements of a sparse matrix M by N are stored into an M x K data array. Rows that do not have K nonzero elements are padded with zeroes. Likewise, column indices are held in another array with a similar padding approach. In addition to this, if the data storing is performed in the form of column-major order, then nearly full memory coalescing is provided on the GPU, which in turn produces higher performance. Moreover, another vector containing the exact number of nonzero elements of each row is also generated. Therefore, unnecessary operations of padded zeroes are prevented with the incorporation of this additional vector into the algorithm. This format is represented in Figure 10. This figure shows the arrays of data, column indices, and nonzero elements per row for a general sparse matrix A.

The results of the constituted conjugate-gradient iterative solver that uses the ELLPACK format are compared with the performance values in another CPU and GPU application from the literature (Layton, Krishnan, & Barba, 2011). In that application, the conjugate-gradient solver of the CUSP library is used for the solution of sparse linear systems, and a performance comparison between a GPU (CUSP) and a CPU implementation is
presented for the solution of sparse linear systems. CUSP is an open-source library for sparse linear algebra and graph computations on CUDA. The CPU and GPU used in that study are Intel Xeon X5650 (32 GB/s bandwidth) and Tesla C2050 (144 GB/s bandwidth), respectively. Moreover, as a compilation option, the basic optimization syntax ‘−O’ is used. Figure 11 shows this comparison for the solution of sparse linear systems. It may be said that a sparse linear system is a good example of bandwidth-limited problems, thus we can expect a speedup value as being the ratio of the bandwidths of the GPU and CPU. In this respect, considering the bandwidths of the GPU (102 GB/s) used in the present study and the CPU (32 GB/s), potential speedup value will be approximately 3.2x for a merely bandwidth problem. Systems in this comparison are formed by the use of a general five-point Poisson stencil. As shown in the figure, the present application of the conjugate-gradient solver yields approximately three times better performance than the application of the CUSP library on the GPU.

6. Conclusions

An implementation of an immersed boundary method on a GPU is performed within the scope of this study. Incompressible Navier-Stokes equations are solved on Cartesian meshes with the purpose of simulating the flows around a square cylinder and a circular cylinder. A direct-forcing approach is utilized for the immersed boundary method. In this approach, the forcing term is inserted into the momentum equation to ensure the proper boundary velocity value on the surface of the immersed body. Moreover, sixth-order, compact, central schemes are used for the discretization of the first and the second spatial derivatives. Time advancement is ensured with the use of a second-order Adams-Bashforth scheme.

Observing the solutions of the flow around a square cylinder, it is seen that the direct-forcing immersed boundary with the compact schemes successfully simulates the wake region and the vortex shedding cycle. However, it is not so successful for the estimation of the force coefficients. One of the reasons for this is spurious oscillations resulting from the quasi-spectral behavior of the compact schemes. Another factor in this context is the sharp corners that somewhat disable the internal flow treatment applied for smoothing the discontinuities on the immersed boundary. Therefore, the flow around a circular cylinder is also simulated to see the effects of the sharp corners on the force calculations. As expected, the estimation of the lift and drag coefficients is accomplished with greater accuracy. As a result, it may be said that issue of fluid penetration that is a potential limitation of direct-forcing immersed boundary method is not tolerated well enough in the case with the sharp corners (square cylinder). As a future work to alleviate this limitation, one thing is to use a finer mesh resolution around the immersed boundary. However, high-order compact scheme should be modified in order to be applied with different mesh resolutions in the domain. In this context, another approach may be the addition of an auxiliary forcing term to satisfy the mass conservation in the vicinity of immersed boundary.

Moreover, a conjugate-gradient iterative method is also implemented on the GPU for the solution of the Poisson equation. A written custom kernel is used for sparse matrix-vector multiplication, which is the decisive part of the algorithm. The ELLPACK sparse matrix storage format is utilized and found to be more efficient in comparison with other formats.

The main reason for this performance increase is that the ELLPACK format allows nearly full memory coalescing and unnecessary operations of padded zeroes are prevented with the incorporation of an additional vector into the algorithm.

A comparison is performed between the present implementation and the other GPU and CPU applications for the sparse linear system solutions. As a consequence, approximately 3x and 20x accelerations are achieved in comparison with the GPU (CUSP) and the CPU, respectively. It should be noted that this speedup value is attained with respect to a single CPU core. Considering that the CPUs are primarily designed for distributed memory computing, a future work may be to compare the performances of CPU clusters and multi-GPU applications. Generally, GPUs present considerable potential as parallel computing devices. With appropriate algorithms and advancements in numerical tools that can effectively exploit the strengths of the GPUs, they may
become the most influential assistants of researchers in dealing with large, complex fluid-flow problems.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**References**

Balaras, E. (2004). Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations. *Computers & Fluids, 33*(3), 375–404. doi:10.1016/S0045-7930(03)00058-6

Bell, N., & Garland, M. (2009). Implementing sparse matrix-vector multiplication on throughput-oriented processors. Proceedings of the Conference on High Performance Computing, Networking, Storage and Analysis, Portland, Oregon. doi:10.1145/1654059.1654078

Bernardini, M., Modesti, D., & Pirozzoli, S. (2016). On the suitability of the immersed boundary method for the simulation of high-Reynolds-number separated turbulent flows. *Computers & Fluids, 130*, 84–93. doi:10.1016/j.compfluid.2016.02.018

Boersma, B. J. (2011). A 6th order staggered compact finite difference method for the incompressible Navier-Stokes and scalar transport equations. *Journal of Computational Physics, 230*(12), 4940–4954. doi:10.1016/j.jcp.2011.03.014

Brandvik, T., & Pullan, G. (2007). Acceleration of a two-dimensional Euler flow solver using commodity graphics hardware. *Proceedings of the Institution of Mechanical Engineers Part C: Journal of Mechanical Engineering Science, 221*(12), 1745–1748. doi:10.1243/09544062MES813FT

Brandvik, T., & Pullan, G. (2008). Acceleration of a 3D Euler solver using commodity graphics hardware. AIAA Paper 2008-607, the American Institute of Aeronautics and Astronautics. doi:10.2514/6.2008-607

Braza, M., Chassaing, P., & Minh, H. H. (1986). Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *Journal of Fluid Mechanics, 165*, 79–130. doi:10.1017/S0022112086000314

Chiu, P. H., Lin, R. K., & Sheu, T. W. H. (2010). A differentially interpolated direct forcing immersed boundary method for predicting incompressible Navier–Stokes equations in time-varying complex geometries. *Journal of Computational Physics, 229*, 4476–4500. doi:10.1016/j.jcp.2010.02.013

Dash, S. M., & Lee, T. S. (2014). Natural convection from inclined square cylinder using novel flexible forcing IB-LBM approach. *Engineering Applications of Computational Fluid Mechanics, 8*(1), 91–103. doi:10.1080/19942060.2014.11015500

Elsen, E., Legresley, P., & Darve, E. (2008). Large calculation of the flow over a hypersonic vehicle using a GPU. *Journal of Computational Physics, 227*(24), 10148–10161. doi:10.1016/j.jcp.2008.08.023

Franke, R., Rodi, W., & Schonung, B. (1990). Numerical-calculation of laminar vortex-shedding flow past cylinders. *Journal of Wind Engineering and Industrial Aerodynamics, 35*(1–3), 237–257. doi:10.1016/0167-6105(90)90219-3

Hagen, T. R., Lie, K.-A., & Natvig, J. R. (2006). Solving the Euler equations on graphics processing units. *Proceedings of the Computational Science – ICCS Pt 4, 220–227. doi:10.1007/11758549_34

Hejranfar, K., & Ezzatneshan, E. (2014). A high-order compact finite-difference lattice Boltzmann method for simulation of steady and unsteady incompressible flows. *International Journal for Numerical Methods in Fluids, 75*(10), 713–746. doi:10.1002/fld.3916

Hoffmann, K. A., & Chiang, S. T. (2000). *Computational fluid dynamics*. Vichita: Engineering Education System.

Iaccarino, G., & Verzicco, R. (2003). Immersed boundary technique for turbulent flow simulations. *Applied Mechanics Reviews, 56*(3), 331–347. doi:10.1115/1.1563627

Jude, D., & Baeder, J. (2016). Extending a three-dimensional GPU RANS solver for unsteady grid motion and free-wake coupling. 54th AIAA Aerospace Sciences Meeting. AIAA SciTech, (AIAA 2016-1811). doi:10.2514/6.2016-1811

Kalita, J. C., & Dass, A. K. (2011). Higher order compact simulation of double-diffusive natural convection in a vertical porous annulus. *Engineering Applications of Computational Fluid Mechanics, 5*(3), 357–371. doi:10.1080/19942060.2011.11015378

Karantasis, K. I., Polychronopoulos, E. D., & Ekaterinaris, J. A. (2014). High order accurate simulation of compressible flows on GPU clusters over software distributed shared memory. *Computers & Fluids, 93*, 18–29. doi:10.1016/j.compfluid.2014.01.005

Kim, D., & Choi, H. (2006). Immersed boundary method for flow around an arbitrarily moving body. *Journal of Computational Physics, 212*(2), 662–680. doi:10.1016/j.jcp.2005.07.010

Lai, M.-C., & Peskin, C. S. (2000). An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics, 160*(2), 705–719. doi:10.1006/jcph.2000.6483

Laizet, S., & Lamballais, E. (2009). Higher-order compact schemes for incompressible flows: A simple and efficient method with quasi-spectral accuracy. *Journal of Computational Physics, 228*(16), 5989–6015. doi:10.1016/j.jcp.2009.05.010

Layton, S. K., Krishnan, A., & Barba, L. A. (2011). *cuIBM – A GPU-accelerated immersed boundary method*. International Conference on Parallel Computational Fluid Dynamics, Barcelona, Spain. Paper retrieved from http://lorenerbarba.com/wp-content/uploads/2013/01/LaytonKrishnanBarba- parcf2011.pdf

Lee, P., & Wolgemuth, C. W. (2016). An immersed boundary method for two-phase fluids and gels and the swimming of Caenorhabditis elegans through viscoelastic fluids. *Physics of Fluids, 28*, 011901. doi:10.1063/1.4938174

Lele, S. K. (1992). Compact finite-difference schemes with spectral-like resolution. *Journal of Computational Physics, 103*(1), 16–42. doi:10.1016/0021-9991(92)90324-R

Liao, C.-C., Chang, Y.-W., Lin, C.-A., & Mcdonough, J. M. (2010). Simulating flows with moving rigid boundary using immersed-boundary method. *Computers & Fluids, 39*(1), 152–167. doi:10.1016/j.compfluid.2009.07.011

Liu, C., Zheng, X., & Sung, C. H. (1998). Preconditioned multigrid methods for unsteady incompressible flows. *Journal of Computational Physics, 139*, 35–57. doi:10.1006/jcph.1997.5859

Liu, X., Zhong, Z., & Xu, K. (2016). A hybrid solution method for CFD applications on GPU-accelerated hybrid HPC platforms. *Future Generation Computer Systems, 56*(Mar.), 759–765. doi:10.1016/j.future.2015.08.002
Luo, K., Zhuang, Z., Fan, J., & Haugen, N. E. L. (2016). A ghost-cell immersed boundary method for simulations of heat transfer in compressible flows under different boundary conditions. *International Journal of Heat and Mass Transfer, 92*(Jan.), 708–717. doi:10.1016/j.ijheatmasstransfer.2015.09.024

Ma, Z. H., Wang, H., & Pu, S. H. (2014). GPU computing of compressible flow problems by a meshless method with space-filling curves. *Journal of Computational Physics, 263*, 113–135. doi:10.1016/j.jcp.2014.01.023

Meinke, M., Schroder, W., Krause, E., & Rister, T. (2002). A comparison of second- and sixth-order methods for large-eddy simulations. *Computers & Fluids, 31*, 695–718. doi:10.1016/S0045-7930(01)00073-1

Mishra, N., & Sanyasiraju, Y.V.S.S. (2012). Exponential compact higher order scheme for steady incompressible Navier-Stokes equations. *Engineering Applications of Computational Fluid Mechanics, 6*(4), 541–555. Retrieved from http://jeacfm.cse.polyu.edu.hk/

Mittal, R., & Iaccarino, G. (2005). Immersed boundary methods. *Annual Review of Fluid Mechanics, 37*, 239–261. doi:10.1146/annurev.fluid.37.061903.175743

Mohd-Yusof, J. (1997). Combined immersed boundaries/B-splines methods for simulations of flows in complex geometries. CTR Annual Research Briefs. NASA Ames/Stanford University. Paper retrieved from https://web.stanford.edu/group/ctr/ResBriefs97/myusofpdf.pdf

Nagarajan, S., Lele, S. K., & Ferziger, J. H. (2003). A robust high-order compact method for large eddy simulation. *Journal of Computational Physics, 191*(2), 392–419. doi:10.1006/jcph.2002.7087

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Kruger, J., Lefohn, A. E., & Purcell, T. J. (2007). A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum, 26*(1): 80–113. doi:10.1111/j.1467-8659.2007.01012.x

Parnaudeau, P., Lamballais, E., Heitz, D., & Silvestrini, J. H. (2004). Combination of the immersed boundary method with compact schemes for DNS of flows in complex geometry. Proceedings of the Fifth International ERCOFTAC Workshop on Direct and Large-Eddy Simulation (pp. 577–581). Munich, Germany. doi:10.1007/978-1-4020-2313-2_61

Perry, A. E., Chong, M. S., & Lim, T. T. (1982). The vortex-shedding process behind two-dimensional bluff-bodies. *Journal of Fluid Mechanics, 116*(Mar), 77–90. doi:10.1017/S0022112082000378

Peskin, C. S. (1972). Flow patterns around heart valves: A numerical method. *Journal of Computational Physics, 10*(2), 252–271. doi:10.1016/0021-9991(72)90065-4

Ren, W. W., Wu, J., Shu, C., & Yang, W. M. (2012). A stream function–vorticity formulation-based immersed boundary method and its applications. *International Journal for Numerical Methods in Fluids, 70*(5), 627–645. doi:10.1002/fld.2705

Robichaux, J., Balachandar, S., & Vanka, S. P. (1999). Three-dimensional floquet instability of the wake of square cylinder. *Physics of Fluids, 11*(3), 560–578. doi:10.1063/1.869930

Saha, A. K., Biswas, G., & Muralidhar, K. (2003). Three-dimensional study of flow past a square cylinder at low Reynolds numbers. *International Journal of Heat and Fluid Flow, 24*(1), 54–66. doi:10.1016/S0142-727X(02)00208-4

Sahu, A. K., Chhabra, R. P., & Eswaran, V. (2009). Two-dimensional unsteady laminar flow of a power law fluid across a square cylinder. *Journal of Non-Newtonian Fluid Mechanics, 160*(2-3), 157–167. doi:10.1016/j.jnnfm.2009.03.010

Saiki, E. M., & Biringen, S. (1996). Numerical simulation of a cylinder in uniform flow: Application of a virtual boundary method. *Journal of Computational Physics, 123*(2), 450–465. doi:10.1016/j.jcp.1996.0036

Sen, S., Mittal, S., & Biswas, G. (2011). Flow past a square cylinder at low Reynolds numbers. *International Journal for Numerical Methods in Fluids, 67*(9), 1160–1174. doi:10.1002/ffd.2146

Seo, J. H., & Mittal, R. (2011). A high-order immersed boundary method for acoustic wave scattering and low-Mach number flow-induced sound in complex geometries. *Journal of Computational Physics, 230*(4), 1000–1019. doi:10.1016/j.jcp.2010.10.017

Sharma, A., & Eswaran, V. (2004). Heat and fluid flow across a square cylinder in the two-dimensional laminar flow regime. *Numerical Heat Transfer Part A-Applications, 45*(3), 247–269. doi:10.1080/10407780490278562

Silva, A. L. F. L. E., Silveira-Neto, A., & Damasceno, J. J. R. (2003). Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method. *Journal of Computational Physics, 189*, 351–370. doi:10.1002/fld.2007.0102-01

Singh, A. P., De, A. K., Carpenter, V. K., Eswaran, V., & Muralidhar, K. (2009). Flow past a transversely oscillating square cylinder in free stream at low Reynolds numbers. *International Journal for Numerical Methods in Fluids, 61*(6), 658–682. doi:10.1002/fld.1979

Sohankar, A., Norberg, C., & Davidson, L. (1997). Numerical simulation of unsteady low-Reynolds number flow around rectangular cylinders at incidence. *Journal of Wind Engineering and Industrial Aerodynamics, 69-71*, 189–201. doi:10.1016/S0167-6105(97)00154-2

Sohankar, A., Norberg, C., & Davidson, L. (1998). Low-Reynolds-number flow around a square cylinder at incidence: Study of blockage, onset of vortex shedding and outlet boundary condition. *International Journal for Numerical Methods in Fluids, 26*(1), 39–56. doi:10.1002/(SICI)1097-0363(19980115)26:1<39::AID-FLD263>3.0.CO;2-P

Su, S.-W., Lai, M.-C., & Lin, C.-A. (2007). An immersed boundary technique for simulating complex flows with rigid boundary. *Computers & Fluids, 36*, 313–324. doi:10.1016/j.compfluid.2005.09.004

Tseng, Y.-H., & Ferziger, J. H. (2003). A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics, 192*(2), 593–623. doi:10.1016/j.jcp.2003.07.024

Tutkun, B., & Edis, F.O. (2012). A GPU application for high-order compact finite difference scheme. *Computers & Fluids, 55*, 29–35. doi:10.1016/j.compfluid.2011.10.016

Uzun, A., Blaisdell, G. A., & Lyrintzis, A. S. (2004). Application of compact schemes to large eddy simulation of turbulent jets. *Journal of Scientific Computing, 21*(3), 283–319. doi:10.1007/s10915-004-1319-0

Wang, H., Zhang, Y., Ma, X., Qiu, J., & Liang, Y. (2016). An efficient implementation of fourth-order compact finite difference scheme for Poisson equation with Drichlet
boundary conditions. *Computers & Mathematics with Applications*, 71(9), 1843–1860. doi:10.1016/j.camwa.2016.02.022

Wang, P., Abel, T., & Kaehler, R. (2010). Adaptive mesh fluid simulations on GPU. *New Astronomy*, 15(7), 581–589. doi:10.1016/j.newast.2009.10.002

Xian, W., & Takayuki, A. (2011). Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster. *Parallel Computing*, 37(9), 521–535. doi:10.1016/j.parco.2011.02.007

Yang, X. L., He, G. W., & Zhang, X. (2010). Large-eddy simulation of flows past a flapping airfoil using immersed boundary method. *Science China-Physics Mechanics & Astronomy*, 53(6), 1101–1108. doi:10.1007/s11433-010-3213-0

Yoon, D.-H., Yang, K.-S., & Choi, C.-B. (2010). Flow past a square cylinder with an angle of incidence. *Physics of Fluids*, 22(4). doi:10.1063/1.3388857

Zhang, H.-Q., Fey, U., Noack, B. R., Konig, M., & Eckelmann, H. (1995). On the transition of the cylinder wake. *Physics of Fluids*, 7(4), 165–189. doi:10.1063/1.868601

Zhang, N., & Zheng, Z. C. (2007). An improved direct-forcing immersed-boundary method for finite difference applications. *Journal of Computational Physics*, 221, 250–268. doi:10.1016/j.jcp.2006.06.012