21st Annual CFD Symposium
August 8 − 9, 2019,
Bangalore.

Performance Enhancement of an Immersed Boundary Method Based FSI Solver Using OpenMP

Chhote Lal Shah[1,*], Dipanjan Majumdar[1], and Sunetra Sarkar[1]

Department of Aerospace Engineering, IIT Madras, Chennia-600036, India
[*] *Communicating Author, E-Mail: lal.chhote.shah@gmail.com*

Abstract. This work presents a high-fidelity in-house Fluid Structure Interaction (FSI) solver developed by combining discrete forcing Immersed Boundary Method (IBM) with a RK-4 based structural solver. Classification of the grid points as fluid, solid and IB points in the IBM framework and the solution of the pressure correction equations are the two most computationally expensive section in the numerical solver. These computational efforts can be significantly reduced by implementing OpenMP techniques. However, the successive over-relaxation (SOR) iterative method used in the serial code is not suitable for OpenMP parallelization as it shows data dependencies from previous iterations. Therefore, the Red-Black (RB) SOR is implemented to avoid the data dependencies.

Keywords: Fluid-structure interaction, Immersed Boundary Method, OpenMP, Red-Black SOR, parallelisation

1 Introduction

Fluid–structure interaction (FSI) is of great relevance to a wide range of real life engineering applications. One of the main difficulties experienced by researchers in numerically simulating these FSI problems is associated with the handling of complex geometry and time dependent movement of the solid boundary. In the conventional approaches, Arbitrary Langrangian Eulerian (ALE) method based flow solver is coupled with a structural solver to study the combined FSI effects [1]. However, these ALE based approaches are not suitable as significant effort is required to regenerate the mesh at every time-step as the solution is advanced. One alternative method to solve these FSI problems efficiently is to incorporate the Immersed Boundary Method (IBM) [2] as flow solver. Instead of direct application of physical boundary conditions at the grid points, in IBM, the presence of a solid boundary inside a fluid is incorporated by imposing constraints to the model governing equations. Therefore, in IBM framework, even a complex geometry problem involving moving boundaries can easily be handled using simple structured grids without the requirement of re-meshing at every time step. The primary advantage of IBM over body-fitted ALE method is that the task of grid generation is greatly simplified which results in optimized CPU cost and ease of implementation.

In the present work the authors have developed an in-house FSI solver in the C++ environment by combining discrete forcing IBM with RK−4 based structural solver. Although the mesh generation process is trivial in IBM framework, the process of classification of the grid points at every time step for a moving boundary problem is computationally intensive process. The solution of the pressure correction equations is also one of the most expensive computational section. However, parallelization of all these computationally expensive processes using OpenMP (Open Multi-Processing) techniques can significantly reduces the computational time. In the Parallelized solver the red-black Successive Over-Relaxation (SOR) [3, 4] algorithm is implemented for solving the set of algebraic equations to avoid data dependencies from previous iteration which is otherwise present in the conventional SOR algorithm. Passive heaving of an elliptical foil with an active pitching actuation in a free stream flow is simulated to test the efficacy of the parallelized code. Simulations are performed with the following iterative algorithms: (a) the traditional SOR, (b) RB-SOR and (c) RB-SOR with OpenMP on a Linux cluster with sixteen dual processor $2.60GHz$ 64 bit Intel Xeons, totalling 32 processors. Details of the structural and flow solver and the parallelization process is discussed next in Section 2; Computational time taken by the three numerical approaches are compared in Section 3; Finally, Section 4 summarizes the present paper with the concluding remarks.

2 Numerical methodology

2.1 Structural solver

An elliptic cylinder is considered to be elastically mounted in the plunging degree of freedom and is allowed to oscillate in a free stream flow. The equation of motion of the center of the elliptic section is given in the non-dimensional form below [5]:

$$\ddot{\bar{y}} + \left(\frac{cD}{MU_\infty}\right)\dot{\bar{y}} + \left(\frac{2\pi f_N c}{U_\infty}\right)^2 \bar{y} = \frac{2}{\pi AR}\frac{\rho}{\rho_b}C_L, \tag{1}$$

where, M is mass per unit span of the elliptic cylinder; $\bar{y}$ denotes the vertical displacement non-dimensionalized by the length of the major axis (D); c is the damping coefficient, U_∞ is the free stream velocity, f_N is the structural natural frequency, AR is the ratio of minor to major axis of elliptic cylinder, ρ/ρ_b is the ratio of fluid to solid density. C_L is the coefficient of lift force acting at the center of the elliptic body and is supplied by a flow solver. An active control is also implemented via a pitching motion as: $\theta = A_\theta sin\,(2\pi f_D \tau)$; where, θ is the angular displacement, A_θ is the pitching amplitude, f_D is the driving frequency and $\tau(= tU_\infty/D)$ is the non-dimensional time. The structural governing equation is solved using the explicit 4^{th} order Runge-Kutta (RK$-$4) method.

2.2 Flow solver

The flow equations are solved broadly following the discrete forcing IBM proposed by Kim *et al.* [2]. In this framework of IBM, the boundary conditions are satisfied at the solid boundaries by adding a additional forcing term to the momentum conservation equation. A mass source/sink term is also added to the continuity equation to ensure the mass conservation across the immersed boundary. The flow equations are solved on a background Eulerian mesh and the location of the solid boundary is tracked by a set of Lagrangian markers. The Eulerian mesh nodes are needed to be classified as fluid points and solid points at every time step. The momentum forcing and the mass source/sink term are applied throughout inside the entire solid domain to reduce the Spurious Force Oscillation (SFO). The flow around the oscillating body is primarily governed by the incompressible Navier-Stokes Equations. The flow governing equations are given in the non-dimensional form below:

$$\frac{\partial \mathbf{u}}{\partial \tau} + \nabla.\,(\mathbf{uu}) = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} + \mathbf{f}, \tag{2}$$

$$\nabla.\mathbf{u} - q = 0, \tag{3}$$

where $\mathbf{u} = (u, v)$ denotes the flow velocity non-dimensionalised by the free stream velocity U_∞, p is the pressure non-dimensionalised with ρU_∞^2 where ρ is the fluid density. $Re = U_\infty D/\nu$ is the Reynolds number where ν is kinematic viscosity of the fluid. Here, $\mathbf{f} = (f_x, f_y)$ is the momentum forcing applied to enforce the no-slip boundary condition at the solid boundary immersed in the fluid.

The flow governing Eqs. (2) and (3) are solved using Fractional Step Method (FSM) on a Cartesian staggered grid arrangement. The convective term of the momentum equation is discretized using Adams-Basforth discretization whereas the diffusion term is discretized using the Crank-Nicolson technique. The velocity is corrected using a pseudo pressure correction term so that the continuity is satisfied at every time step. The discretised form of Eqs. (2) and (3) are as follows:

$$\frac{\hat{\mathbf{u}}^k - \mathbf{u}^{k-1}}{\Delta\tau} = -\nabla p^{k-1} - \frac{3}{2}H\left(\mathbf{u}^{k-1}\right) + \frac{1}{2}H\left(\mathbf{u}^{k-2}\right) + \frac{1}{2Re}L\left(\mathbf{u}^{k-1}\right) + \frac{1}{2Re}L\left(\mathbf{u}^{k-2}\right) + \mathbf{f}^k, \tag{4}$$

$$\nabla^2 \phi^k = \frac{1}{\Delta\tau}\left(\nabla.\hat{u}^k - \hat{q}^k\right), \tag{5}$$

$$\mathbf{u}^k = \hat{\mathbf{u}}^k - \Delta\tau\nabla\phi^k, \tag{6}$$

$$p^k = p^{k-1} + \phi^k - \frac{\Delta\tau}{Re}\nabla^2\phi^k, \tag{7}$$

where $L(*)$ is the discretised form of the Laplace operator and $H(*)$ denotes the discretised form of the convective term, $\hat{\mathbf{u}}^k$ is the intermediate velocity at k^{th} time step, ϕ is the pseudo pressure correction term and $\Delta\tau$ is the time increment. All the spatial derivatives are evaluated using the second-order central difference scheme.

In the present FSI framework, the flow equations are solved to get the flow-field around the body and the aerodynamic loads acting on the body are evaluated at every time step. Then this lift force computed by the flow solver is supplied to the structural equation to compute the position of the body in the next time step. Thus, at every time step, the flow solver and structural solver exchange informations in a staggered manner resulting in a weak coupling FSI solver.

2.3 Parallelization

OpenMP. The most computationally costly sections in the sequential code are: (i) classifications of the grid points as fluid, solid and IB points in the IBM framework and (ii) solution of the pressure correction equations. OpenMP is implemented in these sections along with the other possible places to enhance the overall performance of the solver. OpenMP is an Application Programming Interface (API) that serves a convenient, scalable model to the developers of shared memory parallel applications [6]. It uses explicit direct multi-threaded and shared memory parallelism. OpenMP uses the fork-join model of parallel execution as presented in Fig. 1. OpenMP employs multiple processors to execute the same code and thus accelerates the numerical process, hence decreases computational time. It is necessary to check data dependencies, data conflicts, race conditions and deadlock situations before implementing OpenMP as it may cause erroneous results, if not taken care properly. OpenMP is called in the loop executable using *"#pragma omp parallel"*. OpenMP is also implemented for solving discretized momentum equations and for reading and writing several data files parallelly.

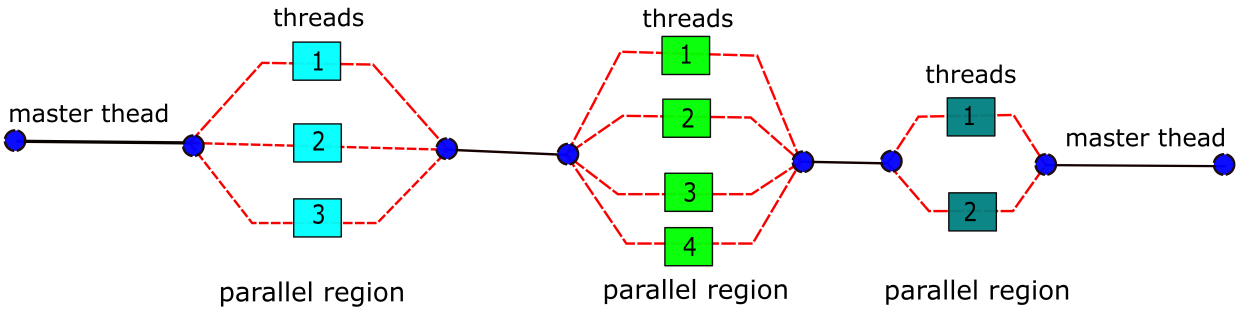

Fig. 1. Fork-Join model for OpenMP parallelisation

Red-Black SOR. The traditional SOR algorithm used in the sequential code suffers from data dependency on previous iteration and therefore cannot be parallelized using OpenMP. The RB-SOR method [3,4], on the other hand, eliminates this issue and takes advantage of OpenMP parallelization. The RB ordering technique and its implementation are schematically shown in Fig. 2. The red and black nodes are located alternatively as shown in the figure. In RB-SOR algorithm, the values at the black nodes are updated using the values at the red nodes and the values at the red nodes are updated using the values at the black nodes. Therefore, it can be partitioned into independent tasks that can be performed by multiple threads simultaneously.

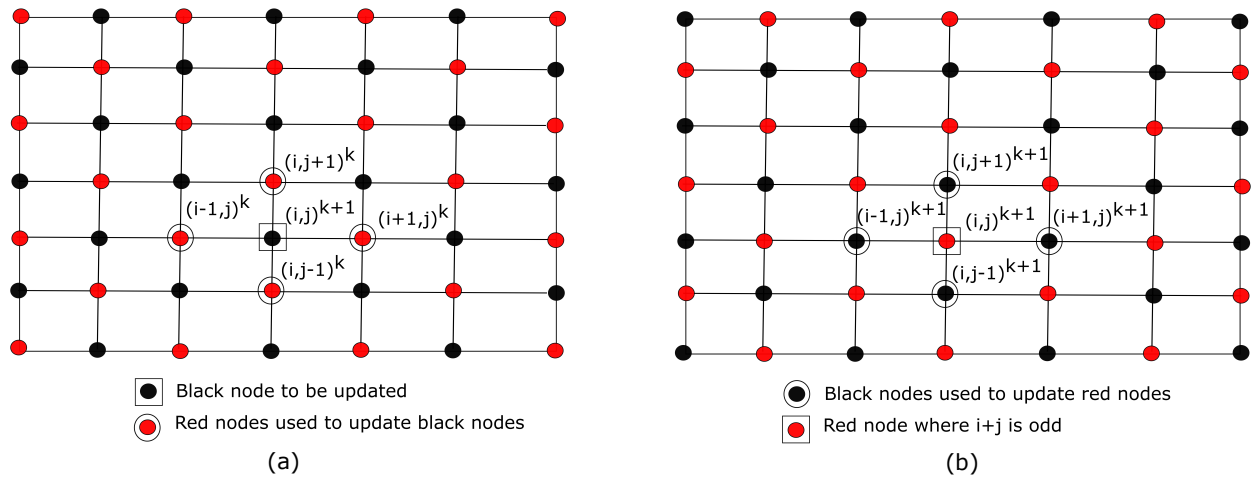

Fig. 2. Red-Black ordering technique and implementation of the SOR algorithm: (a) updating the values of the black nodes using red nodes and (b) updating the values of the red nodes using black nodes.

3 Results and discussion

3.1 Simulation Results

The flow domain and the mesh grid used in the present study are shown schematically in Fig. 3. At the initial time, the centre of the elliptic foil lies at the origin. A Dirichlet condition is applied at the left-hand side boundary as inflow, and the slip boundary conditions are applied at upper and lower boundaries. A Neumann

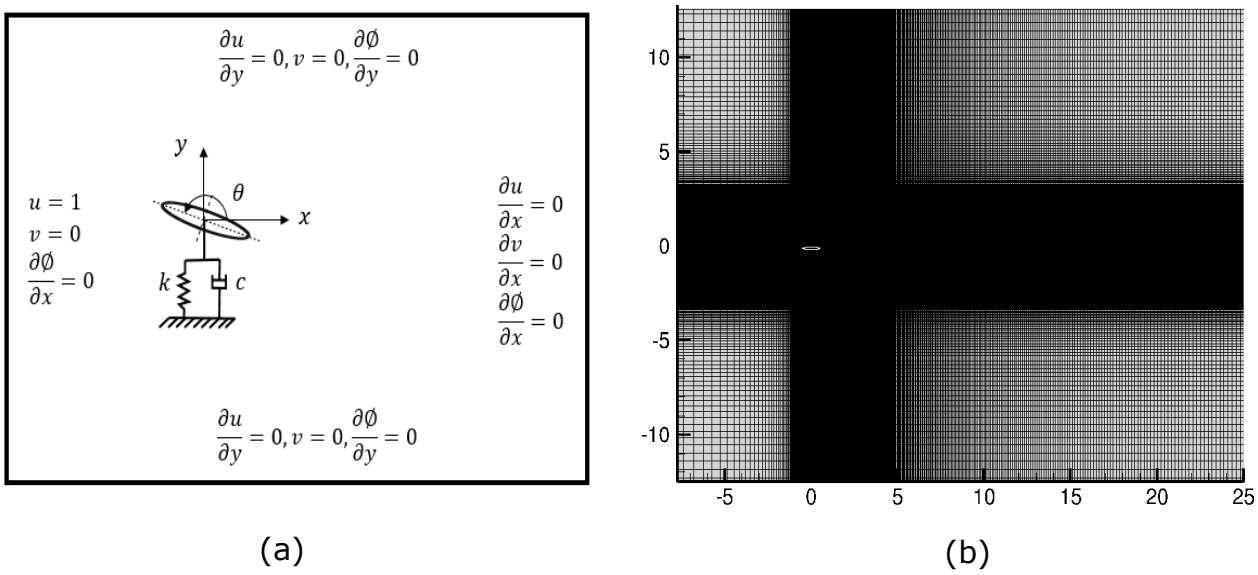

Fig. 3. (a) Flow domain and corresponding boundary conditions and (b) Cartesian grid.

type boundary condition is employed at the velocity outlet. The boundary conditions are shown in Fig. 3(a).

The Non-uniform cartesian grid is considered in this study as shown in Fig. 3(b). For the pressure correction equation, Neumann type boundary condition is applied at all the domain boundaries.

To test the efficacy of the present FSI solver, passive heaving with active pitching actuation of an elliptic cylinder is simulated with parameter values as: $cD/MU_\infty = 0.503$, $f_N c/U_\infty = 0.2$, $AR = 1/6$ and $\rho/\rho_b = 0.2$ and the results are compared with that of Griffth *et al.* [5]. The simulations are performed at a low Reynolds number of $Re = 200$. The lengths of the computational domain along the stream-wise and transverse directions are $32.5D$ and $25D$, respectively. The flow domain is discretized using a structured Cartesian mesh with 718×1017 grid points along the horizontal and vertical directions, respectively. A constant time step of 10^{-4} is considered to perform the simulations. The computational domain, grid spacing and time step size are chosen after performing suitable convergence test. Simulation results of SOR, RB-SOR and RB-SOR with OpenMP solvers match very closely with each other. Here, we present the comparison of instantaneous

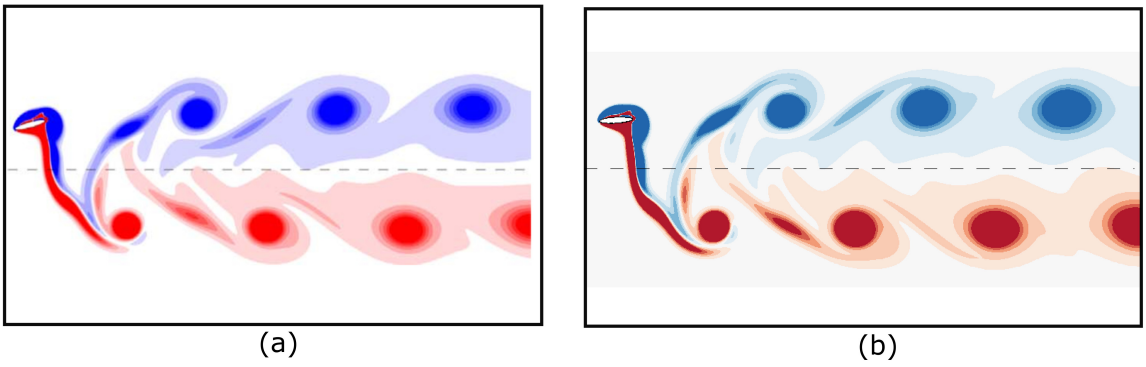

Fig. 4. Comparison of flow-field: (a) Griffith *et al.* [5] and (b) present work using OpenMP parallelisation

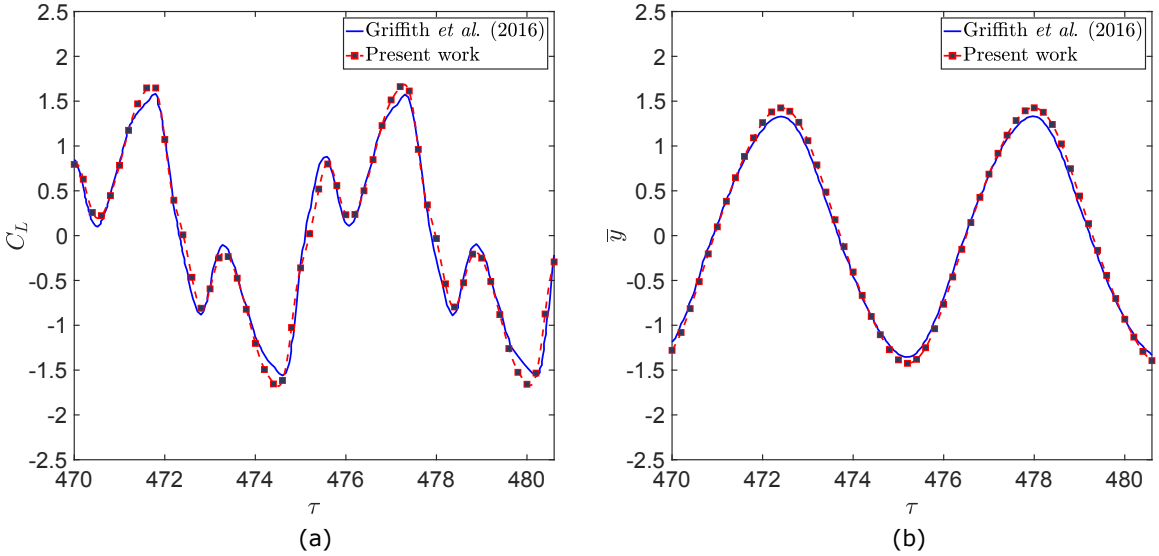

Fig. 5. Comparison of (a) C_L, and (b) vertical displacement y obtained from the present work using OpenMP parallelisation with that of Griffith *et al.* [5].

vorticity contour, evolution lift coefficient (C_L) and vertical displacement ($\overline{y}$) from the present work using OpenMP parallelisation with that of Griffith *et al.* [5] in Fig. 4 and 5, respectively. Very good agreement is observed between the results from the present OpenMP FSI solver and that of Griffith *et al.* [5].

3.2 Performance Results

The performance enhancement of the numerical solvers is evaluated in terms of computational time taken by the solvers to perform 1000 time step iterations *i.e.* $\tau = 0$ to $\tau = 0.1$. The solver speed (S) is defined as

$$S = \frac{T_{SOR} - T_x}{T_{SOR}} \times 100\%, \tag{8}$$

where, T_x indicates the real clock time taken by RB-SOR or RB-SOR with OpenMP. The number of threads considered for the execution of code in parallel are sixteen for all the computations. The performance of the iterative solvers on a Linux cluster with sixteen dual processor $2.60GHz$ 64 bit Intel Xeons, totaling 32 processors is compared in Table 1. The RB-SOR with OpenMP shows significant speedup as compared to sequential FSI solver.

Table 1. Time taken to compute 1000 time steps (non-dimensional time (τ))

Solver	Time taken ($T/sec.$)	Speedup (S)
SOR	12920	–
Red-Black SOR	11780	8.77%
Red-Black SOR with OpenMP	3740	70.61%

4 Conclusion

In this work, a high-fidelity in-house Fluid Structure Interaction (FSI) solver developed by combining discrete forcing Immersed Boundary Method with a RK-4 based structural solver is presented. The RB-SOR method is successfully implemented to get rid of data dependencies from previous iterations and the solver is parallelize using OpenMP technique. The computational time taken to execute 1000 time-steps have been compared between SOR, RB-SOR and RB-SOR with OpenMP parallelization. The RB-SOR with OpenMP parallelization is seen to run significantly faster and has shown the notable speed up as compared to serial code.

References

1. H. T. Ahn and Y. Kallinderis, "Strongly coupled flow/structure interactions with a geometrically conservative ale scheme on general hybrid meshes," *Journal of Computational Physics*, vol. 219, no. 2, pp. 671–696, 2006.
2. J. Kim, D. Kim, and H. Choi, "An immersed-boundary finite-volume method for simulations of flow in complex geometries," *Journal of Computational Physics*, vol. 171, no. 1, pp. 132–150, 2001.
3. C. Zhang, H. Lan, Y. Ye, and B. D. Estrade, "Parallel sor iterative algorithms and performance evaluation on a linux cluster," Naval Research Lab Stennis Space Center MS Oceanography Div, Tech. Rep., 2005.
4. I. Yavneh, "On red-black sor smoothing in multigrid," *SIAM Journal on Scientific Computing*, vol. 17, no. 1, pp. 180–192, 1996.
5. M. D. Griffith, D. L. Jacono, J. Sheridan, and J. S. Leontini, "Passive heaving of elliptical cylinders with active pitching–from cylinders towards flapping foils," *Journal of Fluids and Structures*, vol. 67, pp. 124–141, 2016.
6. B. Chapman, G. Jost, and R. Van Der Pas, *Using OpenMP: portable shared memory parallel programming.* MIT press, 2008, vol. 10.