Proceedings of the ASME 2010 3rd Joint US-European Fluids Engineering Summer Meeting and 8th International Conference on Nanochannels, Microchannels, and Minichannels
FEDSM-ICNMM2010
August 1-5, 2010, Montreal, Canada

FEDSM-ICNMM2010-30149

AN IMMERSED BOUNDARY METHOD BASED ON NAVIER-STOKES EQUATIONS IN DISCRETE STREAM FUNCTION FORMULATION

Xing Zhang *
LNM, Institute of Mechanics
Chinese Academy of Sciences
Beijing, China 100190
Email: zhangx@lnm.imech.ac.cn

Shizhao Wang
LNM, Institute of Mechanics
Chinese Academy of Sciences
Beijing, China 100190
Email: wangsz@lnm.imech.ac.cn

Guowei He
LNM, Institute of Mechanics
Chinese Academy of Sciences
Beijing, China 100190
Email: hgw@lnm.imech.ac.cn

ABSTRACT

A new variant of Immersed Boundary method is proposed in the framework of discrete stream function approach for the Navier-Stokes equations. A parallelized flow solver is developed to simulate two and three-dimensional flow problems involving complex and moving boundaries. The parallel performance of the present flow solver is tested by varying the number of processors used in the simulation. Code validations and applications are also presented, in an order of increasing complexity.

INTRODUCTION

The Immersed Boundary (IB) method is a numerical technique for solving flow problems with complex geometries [1] [2]. This technique has gained popularity recently in the community of computational fluid dynamics due to the great simplification of mesh-generation.

The IB method was first introduced by Peskin [3] to simulate flow interacting with elastic boundaries. This method was later extended to handle rigid boundaries by using the direct forcing technique [4] [5]. In IB methods for incompressible flows, most frequently used Navier-Stokes solvers are based on primitive-variable formulation and projection or SIMPLE-type methods. The stream-function vorticity formulation is seldom used in conjunction with the IB method, the only exception to the best of our knowledge is [6]. However, due to the limitation of the stream function-vorticity formulation itself, it can only be applied to two-dimensional flows. Recently, Colonius and Taira [7] proposed a method that combines the IB method with the discrete stream function (null space) approach to solve Navier-Stokes equations on irregular domains. Although this formulation shares some similarities with that of the stream function-vorticity one, it works well for both two and three-dimensional problems.

In this paper, we proposed an alternative implementation of the IB method based on the discrete stream function approach. We simplify the forcing strategy and reduce both the algorithm complexity and the computational cost. In this new variant of IB method, the cost to compute the force is negligible when compared with that in the basic Navier-Stokes solver. Usually a grid requires several million nodes in a typical three-dimensional simulation where the Re number is in the range of 10^2 to 10^3. When the turnover time of the simulation is considered, feasible computation is severely limited by the current computing power. In order to utilize the computing resources more efficiently, the solver is parallelized using MPI and ported to a cluster. The performance of the code in terms of parallel efficiency is tested by varying the number of processors used.

The outline of the paper is as follows. The numerical methodology is presented in section 2. The code parallelization and efficiency tests are described in section 3. Code validation and application examples are presented in section 4. Finally, the conclusions are drawn in section 5.

*Address all correspondence to this author.

NUMERICAL METHODOLOGY

The three-dimensional incompressible viscous flow is considered in the present work. The governing equations of the flow can be written as:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} + \mathbf{f}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ and p represent velocity vector and pressure respectively. $\mathbf{f}$ is the force representing the appearance of immersed body in the flow. The Reynolds number is defined as $Re = UL/\nu$, where U and L are the characteristic velocity and length respectively, and ν is the kinematic viscosity of the fluid.

The three-dimensional incompressible Navier-Stokes equations (1) and (2) are solved using the discrete stream function (null space) approach [8]. Unlike the classic fractional step method,in this method the divergence-free condition is satisfied to machine precision and there are no splitting errors associated with it. The discretized form of equaitons (1)-(2) can be expressed by a matrix form as

$$\begin{bmatrix} \mathbf{A} & \mathbf{G} \\ \mathbf{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} q^{n+1} \\ p \end{bmatrix} = \begin{bmatrix} r^n \\ 0 \end{bmatrix} - \begin{bmatrix} bc_1 \\ bc_2 \end{bmatrix} + \begin{bmatrix} \tilde{f} \\ 0 \end{bmatrix}, \tag{3}$$

where q, p, and $\tilde{f}$ are the discrete velocity flux, pressure, and body force respectively. The discrete velocity u, can be related to q by multiplying the cell face area. $\mathbf{A}$, $\mathbf{G}$ and $\mathbf{D}$ are the implicit operator, gradient operator and divergence operator respectively. In addition, the negative transpose of the divergence operator is the gradient operator, $\mathbf{G} = -\mathbf{D}^T$. r^n is the explicit right-hand side of the momentum equation. bc_1 and bc_2 are the boundary condition vectors for the momentum and continuity equations.

In the discrete stream function approach [8], a discrete stream-function s is define, such that $q = \mathbf{C}s$, where $\mathbf{C}$ is the curl operator (which is a non-square matrix). This matrix is constructed in such a way that $\mathbf{D}$ and $\mathbf{C}$ enjoy the relation $\mathbf{DC} = \mathbf{0}$, thus the incompressibility condition is automatically satisfied.

The definition in Eqn.(4) together with the relation in Eqn.(5) guarantee the discrete incompressibility. In the discrete stream function approach, another type of curl operator which is called the rotation operator $\mathbf{R}$ is also used. The matrix $\mathbf{R}$ and matrix $\mathbf{C}$ enjoy the relation $\mathbf{R} = \mathbf{C}^T$.

By pre-multiplying the momentum equation with $\mathbf{R}$, the pressure can be eliminated and the system of Eqn. (3) is reduced to a single equation for s at each time step

$$\mathbf{C}^T\mathbf{A}\mathbf{C}s^{n+1} = \mathbf{R}(r^n - bc_1) + \mathbf{R}\tilde{f} = \mathbf{R}r'^n + \mathbf{R}\tilde{f}. \tag{4}$$

The matrix $\mathbf{C}^T\mathbf{A}\mathbf{C}$ is symmetric, positive-definite and thus can be solved using the Conjugate Gradient (CG) method.

As to time advancement, the diffusion term is implicit and treated by the trapezoidal method; the convection term is explicit and a three-step second-order, low storage, Runge−Kutta scheme is used [9].

The forcing term $\mathbf{f}$ on the right hand side of Eqn. (1) is computed in an implicit way. Within one step of time advancing (from n to n+1), this procedure can be summarized as three substeps:

i) A 'predicted' stream function is computed without the forcing and the velocity vectors are reconstructed using the 'predicted' stream function.

ii) A force is applied to achieve the desired velocity on the boundary. In this paper, we follow a similar procedure as that in [10] to determine the force. Mathematically, the forces at the Largrangian points that are used to represent the solid boundary are computed using the formula

$$\sum_{j=1}^{M}\left(\sum_{x}\delta_h(\mathbf{x}-\mathbf{X}_j)\delta_h(\mathbf{x}-\mathbf{X}_k)\Delta s h^2\right)\mathbf{F}(\mathbf{X}_j) = \frac{\vec{U}^{n+1}(\mathbf{X}_k) - \vec{U}^{*}(\mathbf{X}_k)}{\Delta t}, \tag{5}$$

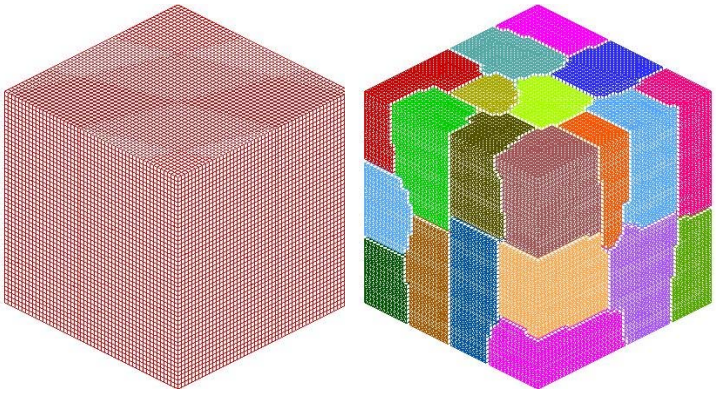
where $\vec{U}^{n+1}$ and $\vec{U}^{*}$ are the desired and 'predicted' velocities respectively; δ_h is the regularized Delta function; h and Δs are the grid sizes of the Euler and Largraigian points respectively; Δt is the time step. By definition, the forces at the (computational) grid points $\mathbf{f}$ can be computed by the transformation

$$\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{M}\mathbf{F}(\mathbf{X_j})\delta_h(\mathbf{x}-\mathbf{X}_k)\Delta s. \tag{6}$$

A 4-point regularized Delta function [1] is used in Eqn. (5) and (6) in all the simulations thereafter. From Eqn. (5), it is seen that to determine the forces, we only need to solve a small linear system in which the number of unknowns is the same as the number of the Largrangian points.

iii) The stream function at time step n+1 is computed with the forcing term computed in ii). The procedure i) - iii) is repeated until the terminating time is reached.

In the work of [7], Eqn.(4) and the non-slip condition on the Largrangian points are reformulated into a larger Karush-Kahn-Tucker (KKT) system; where the forcing term appears as a set of Largrangian multiplier. The LU decomposition (as that in the standard projection method) is then used to derive an equation for $\tilde{f}$. This equation is of Poisson-like where $(\mathbf{C}^T\mathbf{A}\mathbf{C})^{-1}$ appears in the modified Laplacian operator. A 'nested iteration' is needed to solve it directly and the computational cost is very large. In this paper, we propose to employ a forcing strategy that is simple and straightforward. The implementation of this strategy is also very easy. Since the number of unknowns of the linear system in



a) b)

FIGURE 1. A SCHEMATIC REPRESENTATION OF MESH PARTITIONING IN THE PREPROCESSING STAGE, a) THE ORIGINAL MESH; b) THE SUB-DOMAINS AND MESHES GENERATED USING METIS.

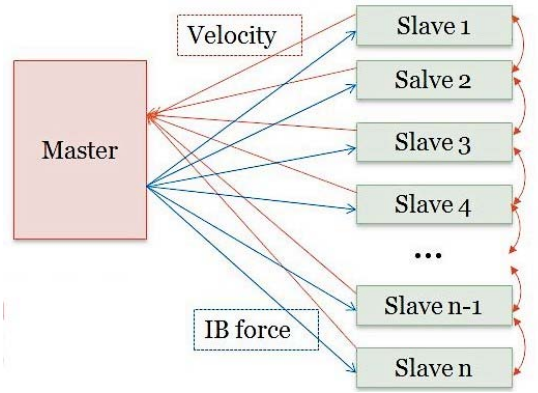


FIGURE 2. A SCHEMATIC DIAGRAM OF THE "MASTER-SLAVE" PARALLEL MODE.

Eqn.(5) is much smaller compared to that of the Navier-Stokes equation, the extra computational cost is negligible.

PARALLEL IMPLEMENTATION

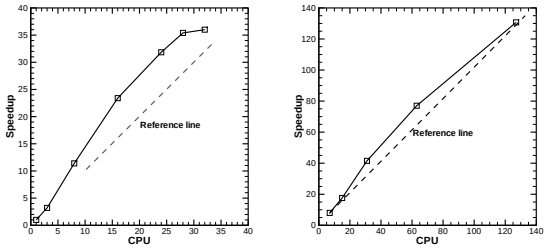
To facilitate an efficient use of the computer resources in distributed-memory architectures, domain decomposition methodology is employed to parallelize the code. Message Passing Interface (MPI) library routines are used for data communication in a master-slave algorithm. An unstructured data structure is used to store the connectivity of the grid that is used in the solver. Grid partition is performed in a preprocessing stage. A well known graph partitioning software METIS [11] is used to partition the grid. To balance the computations among the processors, the partition is done so that the number of cells in each sub-domain is almost the same. To reduce the communication cost between processors, the number of adjacent cells assigned to different sub-domains is minimized. Figure 1 shows the schematic representation of the original mesh before partition and meshes of each sub-domain after partition.

In the parallel implementation, the master process performs the input-output, global time-step control and also the computation of forces. The slave processes solve the flow field within the partition. The flow variables at the partition boundaries are exchanged among the neighboring partitions at each time step. The concept of overlapped communication and computation is used in this implementation to increase parallel efficiency. The computation of the forces which mimic the presence of immersed boundary further complicates the parallel implementation. Keeping in mind the master-slave strategy employed in the parallelization of flow solver, it is reasonable to have a given processor deal with those Largrangian points which are currently located within its local sub-domain. However, such implementation will cause a variety of complications in problems that involve a freely moving boundary. For example, it is necessary to hand-over the control of some Largrangian points from one processor to another processor if the moving boundary crosses the partition border. To overcome this difficulty, we employ an alternative strategy which is called the 'gathering-and-scattering' method. A schematic representation of this method in shown in Fig. 2. In this implementation, some (very limited) global information (such as the position and fluid velocity at the grid points) are sent to and stored in the master at each time step. The master gathers from the slaves the information that are needed, compute the forces and then scatters them to the slaves. The slaves only solve the Navier-Stoke equations using the local forces within the partition. This 'gathering-and-scattering' strategy is very easy to programm. The disadvantage of this method is that the redundant storage of information in the master process causes a waste in memory usage. Furthermore, frequent transfer of data between the master and slaves can jeopardize the efficiency of the code if the number of the grid points is very large. These side-effects can be alleviated by setting a small 'window' that encompasses all the grid points which interact with the Largrangian points. For the grid points in each sub-domain, only those lie within the 'window' are activated in the 'gathering-and-scattering' operations. If the number of Largrangian points is very large (more than 10^4), a parallel solution of Eqn. (5) may also enhance the parallel efficiency.

The index of speedup is used to quantify the parallel performance of the code. This is as a measurement of relative speed of a multiprocessor system and a single processor system and is defined as

$$S(N,P) = \frac{t_s}{t_p}, \tag{7}$$

where N is the size of the problem and P is the number of pro-



a) b)

FIGURE 3. THE SPEEDUP OF THE PARALLEL IMMERSED BOUNDARY FLOW SOLVER. A) 1.4×10^5 UNKNOWNS RUNNING ON 1-32 CPUS. B) 1.2 MILLION UNKNOWNS RUNNING ON 8-128 CPUS.

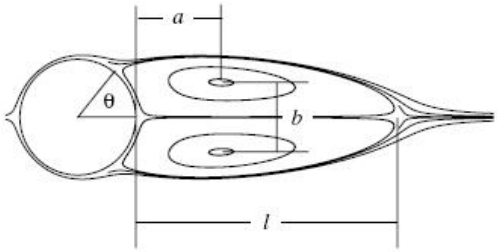


FIGURE 4. THE RECIRCULATION ZONE BEHIND A CYLINDER AT Re = 40. l IS THE LENGTH OF THE RECIRCULATION , a IS THE DISTANCE FROM THE CYLINDER TO THE CENTER OF THE WAKE VORTEX, b IS THE GAP BETWEEN THE CENTERS OF THE WAKE VORTICES, AND θ IS THE SEPARATION ANGLE.

cessors used; t_s is the execution time on a single processor and t_p is the execution time on a multiprocessor.

The three-dimensional lid-driven cavity problem serves as the test case to assess the parallel performance of the code. A Reynolds number of $\mathrm{Re} = UD/\nu = 100$ is simulated for two cases described below, where U is the lid sliding velocity and D is the dimension of the cavity. In case *I* and case *II*, the domain of D × D × D is discretized using a hexahedral mesh with 1.4×10^5 and 1.2 million grid points. To study the performance of the parallel code in obtaining the solution, the domain is divided into N sub-domains, where $1 \leq N \leq 32$ is considered in case *I* and $8 \leq N \leq 128$ in case *II*. We run the code for 200 time steps in both cases and record the wall-clock time. In case *II*, running the code on a single node is not feasible due to the memory size limit. Thus in evaluating the speedup, the execution time on a single processor t_s is replaced by the execution time on 8 CPUs. As shown in Fig. 3, good parallel scalability of the solver is exhibited in both cases. The super linear speedup in the two cases is probably due to cache effect of the computer nodes and further investigations are needed to clarify the situation.

VALIDATIONS AND APPLICATION

To validate the parallel immersed boundary flow solver, some benchmark flows, including the viscous flow around a circular cylinder and a sphere are simulated. The flow around a fish-like body is also simulated to demonstrate the capability of the solver in handling complex and moving boundaries.

Flow around a Cylinder

Viscous flow around a circular cylinder is simulated in a rectangular domain $L \times H = 60D \times 40D$, where D is the diameter of the cylinder. The flow is uniform with velocity U at the inlet. The boundary condition on the surface of the cylinder is non-slip and at the outlet is convective. The Reynolds number Re is defined based on the cylinder diameter D, free-stream velocity U and kinematic viscous coefficient ν. In addition, the drag and lift coefficients are defined as

$$C_D = F_x/(2\rho U^2), \tag{8}$$

$$C_L = F_y/(2\rho U^2). \tag{9}$$

For the simulation of Re = 40, the grid size in the region of $2D\times2D$ near the cylinder is 0.033D. For the simulation of Re = 100, the grid size in the region of $1.5D\times1.5D$ near the cylinder is 0.02D. The grids are stretched to the boundaries with a factor of 1.05 and the maximum grid size in both cases is 0.5.

At Re = 40, the flow is steady and the recirculation zone is characterized by the length l, the distance from the back of cylinder to the vortex center a, the distance between to two vortex centers b, and the separation angle θ, as defined in Fig. 4. The results of the present computation and the data from the literatures are listed in Tab. 1. The streamlines and the contours of vertical velocity component are shown in Fig. 5. At Re = 100, the flow is characterized by alterative vortex shedding from the cylinder. The instantaneous vorticity contours are shown in Fig. 6. The comparison of the present results (drag and lift coefficients and Strouhal number) with those from the literatures is presented in Tab. 2. For both cases (Re = 100 and 200), the numerical results are in good agreements with those from the literatures.

Flow around a Sphere

Viscous flow around a sphere are investigated at Re = 100 and Re = 300 respectively. Here the Reynolds number Re is defined based on the free stream velocity U, the diameter of the

TABLE 1. DRAG COEFFICIENT AND CHARACTERISTICS OF RECIRCULATION ZONE FOR FLOW PAST A CYLINDER AT Re = 40

Case	l/D	a/D	b/D	θ	C_D
Present	2.36	0.72	0.6	53.8°	1.54
Dennis and Chang [12]	2.35	-	-	53.8°	1.52
Linnick and Fasel [13]	2.28	0.72	0.6	53.7°	1.54
Taira and Colonius [14]	2.33	0.72	0.6	53.6°	1.54

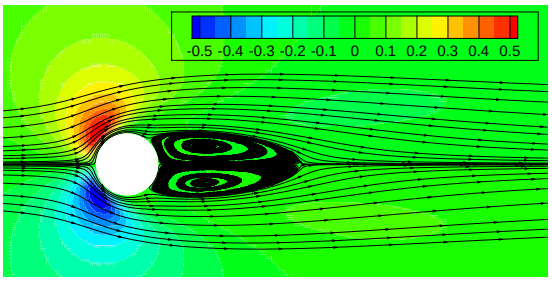


FIGURE 5. STREAMLINES AND CONTOURS OF VERTICAL VELOCITY COMPONENT FOR FLOW PAST A CYLINDER AT Re = 40.

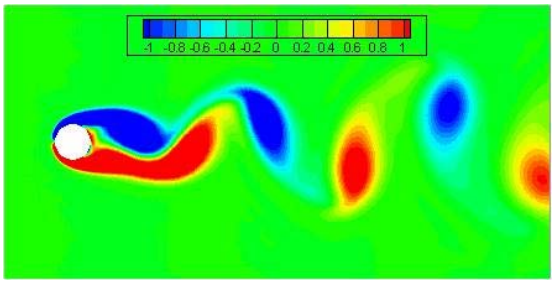


FIGURE 6. INSTANTANEOUS VORTICITY CONTOURS FOR FLOW PAST A CYLINDER AT Re = 100 .

sphere D and the kinematic viscosity ν. The computation is performed in a domain of $L \times H \times W = 30D \times 30D \times 30D$. The drag and lift coefficients are defined as

$$C_D = F_x/(0.5\rho U^2 \pi D^2/4), \tag{10}$$

$$C_L = F_y/(0.5\rho U^2 \pi D^2/4). \tag{11}$$

TABLE 2. DRAG AND LIFT COEFFICIENTS AND STROUHAL NUMBER FOR FLOW PAST A CYLINDER AT Re = 100.

Case	C_D	C_L	St
Present	1.32	$\pm$0.32	0.166
Liu et al [15]	1.35	$\pm$0.32	0.164
Park et al [16]	1.33	$\pm$0.33	0.165

TABLE 3. DRAG COEFFICIENT FOR FLOW PAST A SPHERE AT Re = 100

Case	C_D
Present	1.13
Johnson and Patel [17]	1.10
Fadlun et al [4]	1.08

For the case of Re = 100, a mesh with the uniform size of 0.025 is used in the region of $1.5D \times 1.5D \times 1.5D$ surrounding the sphere and the total number of nodes is approximately of 1 million. For the case of Re = 300, a mesh with the uniform size of 0.0125 is used in the region of $1.25D \times 1.25D \times 1.25D$ surrounding the sphere and the total number of nodes for this case is 1.2 million. The uniform velocity of U is prescribed on the inlet. The boundary condition on the surface of the sphere is non-slip, and at the outlet is convective.

At Re = 100, the the flow is steady and axisymmetric with a separation bubble behind the sphere. The drag coefficient of the present calculation is compared with the data in the literatures in Tab. 3. At Re = 300, the flow exhibits unsteady characteristics such as vortex shedding and oscillations of drag and lift coefficients. The vortical structures are shown in Fig. 7, where the vortical surfaces are identified using the method by Jeong and Hussain [18]. The averaged drag and lift coefficients are shown in Tab. 4. The results for Re = 100 and Re = 300 agree well with those from the literatures.

Flow around a Fish-like Body

The swimming of a fish-like body is simulated using the present immersed boundary flow solver. A body shape representing a tuna is employed. The parameters of the model are taken from the work of Zhu et al [20] with a little modification. The wave-like motion of body is described as (in the body-fixed

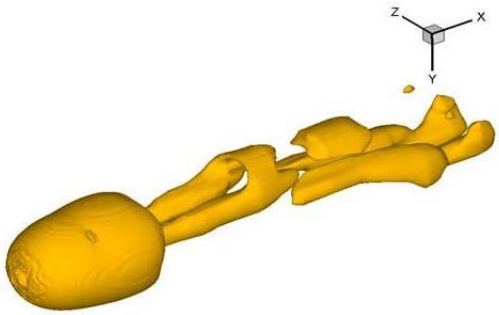


FIGURE 7. INSTANTANEOUS VORTICAL STRUCTURES FOR FLOW PAST A SPHERE AT Re=300.

TABLE 4. DRAG AND LIFT COEFFICIENTS AND STROUHAL NUMBER FOR FLOW PAST A SPHERE AT Re = 300.

Case	C_D	C_L	St
Present	0.68	0.071	0.135
Johnson and Patel [17]	0.66	0.069	0.137
Kim et al [19]	0.66	0.067	0.134

coordinate system)

$$\begin{aligned} y(x,t) &= a(x)\sin(k_\omega x - \omega t), \\ a(x) &= c_1 x + c_2 x^2, \end{aligned} \tag{12}$$

where $k_\omega \equiv 2\pi/\lambda$ is the wave number, corresponding to wavelength λ , ω is the circular frequency of oscillation, $a(x)$ is defined by the adjustable parameters c_1 and c_2 , representing the smoothly varying of the amplitude of the waving body.

The caudal fins of the fish-like body are allowed to undergo a pitch motion around the front point. The pitching motion is taken as:

$$\theta = \alpha \sin(k_w x_p - \omega t - \phi) \tag{13}$$

where α is the amplitude of the pitch angle and ϕ the phase lag between the undulation and pitch.

The parameters in present simulation are: $\lambda = 1.22$, $\omega = 2\pi$, $c_1 = 0.0$, $c_2 = 0.155$, $\alpha = \pi/6.0$, $\phi = \pi/2.0$. The Reynolds number Re based on the free stream velocity U, the length of the fish-like body L and the kinematic viscosity ν in the present simulation is 1000.

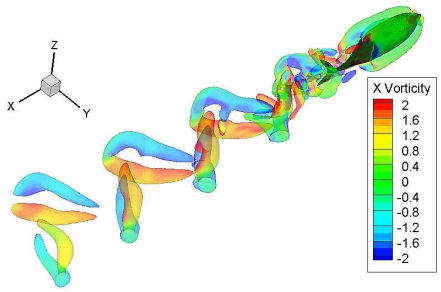


FIGURE 8. WAKE STRUCTURES IN THE SWIMMING OF A FISH-LIKE BODY.

The wake structures of the swimming of fish-like body are shown in Fig. 8. The vortical structures in the wake are visualized using the iso-surfaces of the second invariant of the velocity-gradient tensor. The iso-surfaces are colored with the values of stream-wise vorticity. The structures are arranged in a single row and remain confined within a relatively narrow strip that is centered about the horizontal axis of the fish body. The wake structures consist of a series of hairpin-like vortices braided together such that the legs of each vortex are attached to the head of the preceding one.

CONCLUSIONS

In this paper, we report the development of an Immersed Boundary flow solver in combination with the discrete stream function approach for incompressible Navier-Stokes equations. The parallel implementation of the code on a distributed-memory cluster is also described. The parallel performance tests show a super linear behavior in the speedup of the code. The code is validated by some canonical problems such as the flows over a cylinder and a sphere. The results are in agreement with the data from other references. To demonstrate the capability of the code in simulating three-dimensional flows that involve complex and moving boundaries, the swimming of a fish-like is also studied.

ACKNOWLEDGMENT

This work was supported by Chinese Academy of Sciences under the Innovative Project ‘Multi-scale modeling and simulation in complex systems’ (KJCX-SW-L08), ‘Mathematical modeling of complex system’(KJCX3-SYW-S01), National Basic Research Program of China (973 Program) under Project No. 2007CB814800, and National Natural Science Foundation of China under Project Nos. 10702074 and 10872201.

REFERENCES

[1] Peskin, C., 2002. "The Immersed Boundary Method". *Acta. Numerica.,* ***11***, pp. 479–517.

[2] Mittal, R., and Iaccarino, G., 2005. "Immersed Boundary Methods". *Annu. Rev. Fluid Mech.,* ***37***, pp. 239–261.

[3] Peskin, C., 1972. "Flow Patterns around Heart Valves: A Numerical Method". *J. Comput. Phys.,* ***10***, pp. 252–271.

[4] Fadlun, E., Verzicco, R., Orlandi, P., and Mohd-Yusof, J., 2000. "Combined Immmersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations". *J. Comput. Phys.,* ***161***, pp. 35–60.

[5] Iaccarino, G., and Verzicco, R., 2003. "Immersed Boundary Technique for Turbulent Flow Simulations". *Appl. Mech. Rev.,* ***56***, pp. 331–347.

[6] Wang, Z., Fan, J., and Cen, K., 2009. "Immersed Boundary Method for the Simulation of 2D Viscous Flow Based on Vorticity-Velocity Formulations". *J. Comput. Phys.,* ***228***, pp. 1504–1520.

[7] Colonius, T., and Taira, K., 2008. "A Fast Immersed Boundary Method using a Nullspace Approach and Multi-domain Far-Field Boundary Conditions". *Comput. Methods Appl. Mech. Engrg.,* ***197***, pp. 2131–2146.

[8] Chang, W., Giraldo, F., and Perot, B., 2002. "Analysis of an Exact Fractional Step Method". *J. Comput. Phys.,* ***180***, pp. 183–199.

[9] Perot, B., and Nallapati, R., 2003. "A Moving Unstructured Staggered Mesh Method for the Simulation of Incompressible Free-Surface Flows". *J. Comput. Phys.,* ***184***, pp. 192–214.

[10] Su, S., Lai, M., and Lin, C., 2007. "An Immersed Boundary Technique for Simulating Complex Flows with Rigid Boundary". *Comput. Fluids,* ***36***, pp. 313–324.

[11] Karypis, G., and Kumar, V., 1999. "A Fast and High Quality Scheme for Partitioning Irregular Graphs". *SIAM J. Sci. Comput.,* ***20***, pp. 259–392.

[12] Dennis, S., and Chang, G., 1970. "Numerical Solutions for Steady Flow Past a Circular Cylinder at Reynolds Number up to 100". *J. Fluid Mech.,* ***42***, pp. 471–489.

[13] Linnick, M., and Fasel, H., 2005. "A High-Order Immersed Interface Method for Simulating Unsteady Incompressible Flows on Irregular Domains". *J. Comput. Phys.,* ***204***, pp. 157–192.

[14] Taira, K., and Colonius, T., 2007. "The Immersed Boundary Method: A Projection Approach". *J. Comput. Phys.,* ***225***, pp. 2118–2137.

[15] Liu, C., Sheng, X., and Sung, C., 1998. "Preconditioned Multigrid Methods for Unsteady Incompressible Flows". *J. Comput. Phys.,* ***139***, pp. 35–57.

[16] Park, J., Kwon, K., and Choi, H., 1998. "Numerical Solutions of Flow Past a Circular Cylinder at Reynolds Numbers up to 160". *KSME Int. J.,* ***12***, pp. 1200–1205.

[17] Johnson, T., and Patel, V., 1999. "Flow Past a Sphere up to a Reynolds Number of 300". *J. Fluid Mech.,* ***378***, pp. 19–70.

[18] Jeong, J., and Hussain, F., 1995. "On the Identification of a Vortex". *J. Fluid Mech.,* ***285***, pp. 69–64.

[19] Kim, J., Kim, D., and Choi, H., 2001. "An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries". *J. Comput. Phys.,* ***171***, pp. 132–150.

[20] Zhu, Q., Wolfgang, M., and Triantafyllou, M., 2002. "Three-Dimensional Flow Structures and Vorticity Control in Fish-like Swimming". *J. Fluid Mech.,* ***468***, pp. 1–28.