An efficient tree-topological local mesh refinement on Cartesian grids for multiple moving objects in incompressible flow

Wei Zhang[a], Yu Pan[a], Junshi Wang[a,1], Valentina Di Santo[b], George V. Lauder[c], Haibo Dong[a,*]

[a] *Department of Mechanical and Aerospace Engineering, University of Virginia, 122 Engineer's Way, Charlottesville, 22904, Virginia, USA*
[b] *Division of Functional Morphology, Department of Zoology, Stockholm University, Svante Arrhenius väg 18B, Stockholm, SE-11419, Sweden*
[c] *Department of Organismic and Evolutionary Biology, Harvard University, 26 Oxford Street, Cambridge, 02138, Massachusetts, USA*

Abstract

This paper develops a tree-topological local mesh refinement (TLMR) method on Cartesian grids for the simulation of bio-inspired flow with multiple moving objects. The method solves the time-dependent incompressible flow using a fractional-step method and discretizes the Navier-Stokes equation using a finite-difference formulation with an immersed boundary method to resolve the complex boundaries. The discretized equations are solved iteratively on the refinement mesh with ghost-cell communication between blocks, therefore enabling parallel computation on a distributed memory system. For better accuracy and faster convergence, the momentum equation is solved on non-overlapped refinement meshes, while the Poisson equation is solved on overlapped meshes, recursively from the coarsest block to the finest ones, or parallelly using the Schwarz method if child blocks of the same tree node are connected. Convergence studies show that the algorithm is second-order accurate in space for both velocity and pressure, and the developed mesh refinement technique is benchmarked and demonstrated by several canonical flow problems. The TLMR enables a fast solution to an incompressible flow problem with complex boundaries or multiple moving objects. Various bio-inspired flows of multiple moving objects show that the solver can save over 80% computational time, proportional to the grid reduction when refinement is applied.

Keywords: local mesh refinement, tree topology, bio-inspired flow, immersed boundary method, distributed memory

1. Introduction

Bio-inspired flow dynamics studies the external flow stirred by insects, birds, or fishes, or flow inside organs of humans or animals. It has a broad range of applications in biomimetic engineering and human health studies [1–4]. Different from canonical flow simulation problems, bio-inspired flow features unsteady flow restrained by complex boundaries, such as flexible or

*Corresponding Author: hd6q@virginia.edu
[1] Present address: Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ 08544

moving surfaces. Despite the successes of moving unstructured meshes used in a finite volume or a finite element method [5–7], Cartesian grids with immersed boundary (IB) methods [8–10] are one of the most popular approaches for studying cases with complex boundaries. The IB method solves the Navier-Stokes equations on a fixed Cartesian grid and models the solid boundaries by a field of forces to enforce the boundary conditions and therefore enables an effective solution for moving boundary problems. The success of the IB method has attracted a great deal of research interest [11–16]. The sharp-interface immersed boundary method using a direct forcing approach achieved great success in various bio-inspired flows with complex and moving boundaries [17–20]. Despite its enormous success, the IB method poses challenges to computing resources because of the large number of meshes required for a smooth representation of complex boundaries. This issue can be even more demanding in flow with multiple moving objects (MMO), such as a flock of flying birds or schooling fish, which is often characterized by a large computational domain with highly non-homogeneous grid resolutions. Hence, the design of a fast and efficient technique for such problems is an urgent need.

The local mesh refinement (LMR), or the local adaptive mesh refinement (AMR) techniques [21, 22], which locally refine the mesh without significantly increasing the total number of meshes, may mitigate the demands for computing resources and provides an attractive solution. The subdivision and the addition of new elements usually change the data structure. Hence, the unstructured meshes [23, 24] or the more advantaged data structures, such as the tree-based multi-layer grids [25], are commonly used in various AMR techniques. However, the IB method may prefer simple structured Cartesian grids, like the multi-layer block-structured grids used by Berger et al. [26] for hyperbolic systems. In their approach, a sequence of blocks containing finer Cartesian grids will be automatically generated or removed by evaluating a user-specified error function until the solution is sufficiently resolved. This technique achieved substantial success on various two-dimensional (2D) and three-dimensional (3D) hyperbolic systems or compressible flows [27–30].

The block-based AMR techniques have also been integrated with IB methods to solve the incompressible flow problem and some popular patch-based or octree-based (for 3D problem) refinement techniques have emerged [31–35]. Unlike compressible flow problems, which can be numerically advanced in time, for incompressible flows the elliptic Poisson equation needs to be solved to enforce a divergence-free velocity field. One concern is computational efficiency, such as computational time or memory, of the Poisson equation under such mesh refinement techniques, especially when the refinement contains too many small refinement blocks and parallel computation on a distributed memory is required. Moreover, the AMR technique often allocates lots of computational resources to resolve the far wake, while the bio-inspired flow simulation usually focuses on near-wake-region properties, such as drag and lift forces or the interactions between fluid and structures. The allocation and deallocation of refinement meshes may incur an additional overshoot in computational load in the simulations. Peng et. al [36] demonstrated that with a few nested blocks containing Cartesian grids they can improve the discretization accuracy around the solid boundaries. Their approach is based on the finite volume approach and was applied only to 2D flow problems with stationary boundaries. Deng and Dong [37] presented an octree-like local mesh refinement technique for bio-inspired flow simulation with enhanced computation ability due to a reduced number of meshes. Recently, Zhang et. al [38] developed a block-based mesh refinement technique using a finite-difference formulation with IB method to simulate human airway flow with a moving uvula. Direct application of the aforementioned nested Cartesian grids to 3D flows with MMO is difficult and the computation of the Poisson equations on multiple refinement blocks on a distributed memory system is yet to be addressed.

This paper develops a TLMR with IB method embedded (TLMR-IBM) flow solver for bio-inspired flow with MMO. This method recursively refines local mesh without significantly increasing the number of meshes and can be applied to highly non-homogeneous flow with MMO. More importantly, an effective iterative procedure is developed on these TLMR meshes for a fast solution of the discretized momentum and continuity equations. With the proposed TLMR method, an existing Cartesian-grid-based flow solver can be readily adapted and parallelized. This paper is organized as follows. § 2 introduces the proposed mesh refinement technique and an iterative procedure to solve the discretized Navier-Stokes equation is also presented for such meshes. § 3 presents some benchmark examples and demonstrates the accuracy and efficiency of the proposed mesh refinement technique. Analyses of other complicated flows past MMO are also demonstrated to illustrate the application of this approach.

2. Numerical methods

This section introduces the meshes for TLMR and the procedures to integrate the incompressible Navier-Stokes equations on such meshes.

2.1. The meshes for TLMR

The meshes for TLMR can be introduced using the example in Fig. 1(a), where a school of fish is swimming. To simulate flow around such a group, a dense mesh is required around each swimmer. Instead of having a uniform mesh for the computational domain, we can gradually refine the mesh with refined blocks. For the problem illustrated in Fig. 1(a), one block with the background Cartesian grids covers the whole computational domain. Then another block covers the school with an additional refinement block around each fish for better resolution. To better resolve the body-body/fin interaction between different sizes of fishes, one more refinement block can be placed around the smaller fish. If the wake is of interest, an additional refinement block is placed in the far wake region.

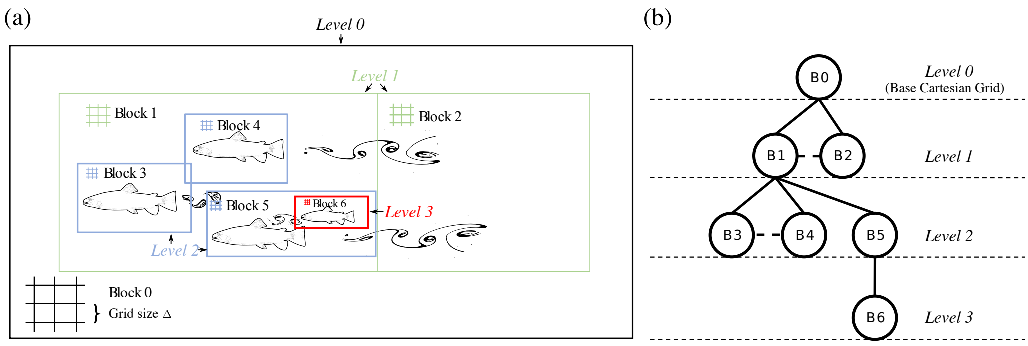

Figure 1: Schematic of TLMR for flow with multiple moving objects: (a) a bio-inspired flow problem with local mesh refinement and (b) a tree topology for the refinement blocks, with solid lines denoting interlayer connections and dash lines for intralayer connections.

The above mesh refinement has nested the fine blocks inside a coarse one. This approach balances the need for desired resolution around solid boundaries and the simplicity of communication between refinement blocks. The parent and the child hierarchy of refinement blocks resembles a tree topology, which can be employed to describe the connectivity between these blocks. In this description, as referenced in Fig. 1(b), each refinement block is a node of the tree and its refinement block is its child node. We may further require that each node can have only one parent block, meaning that a refinement block is located inside a coarse one. This restriction may degrade slightly the flexibility of adding refinement blocks but can greatly simplify information communication between the blocks because of the simplified connection. The child blocks under the same parental node may also be connected, such as block 1 (B1)-B2 or B3-B4 in Fig. 1(b). An additional dashed line is added for such intralayer-connected blocks. The boundary-induced mesh refinement for bio-inspired flow simulations often adopts a fixed hierarchical mesh refinement and does not need to be changed during the simulation. These predetermined meshes can avoid the overhead of dynamic allocation and deallocation of grids in a standard AMR technique.

Meshes in the refined block are obtained by subdividing that of the coarse block in each direction by a factor of two. Hence, a 3D cell will have eight subcells or four for a 2D case. By adjusting the resolution of background meshes and the total levels of refinements, the local refinement approach can provide the necessary grid resolution without significantly increasing the overall number of meshes.

2.2. Fractional-step method for incompressible Navier-Stokes equations with immersed boundaries

The bio-inspired flow is usually described by the unsteady incompressible Navier-Stokes equations

$$\frac{\partial u_i}{\partial t} + \frac{\partial \left(u_i u_j\right)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{Re}\frac{\partial^2 u_i}{\partial x_j x_j}, \tag{1}$$

$$\frac{\partial u_i}{\partial x_i} = 0,, \tag{2}$$

where $i, j = 1, 2$, or 3, and u_1, u_2, and u_3 are the dimensionless velocity in x-, y-, and z-direction respectively, and p is the dimensionless pressure of the fluid. Re is the Reynolds number.

The incompressible Navier-Stokes equations are discretized using a cell-centered, collocated arrangement of the primary variables u_1, u_2, u_3, and p. The coupled system of velocity and pressure is integrated in time using the fractional-step method [39, 40], where it first computes an approximation solution $\boldsymbol{u}^*$ to the momentum Eqn. (1) by

$$\frac{u_i^* - u_i^n}{\Delta t} = \frac{1}{2}(3N_i^n - N_i^{n-1}) + \frac{1}{2Re}(\frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta y^2} + \frac{\delta^2}{\delta z^2})(u_i^n + u_i^*), \tag{3}$$

using a second-order Adams-Bashforth scheme for the convective terms and an implicit Crank-Nicolson scheme for the viscous term to eliminate the viscous stability constraint. Nonlinear convective terms are represented as $N_i = -\delta(u_i u_j)/\delta x_j$. $\delta/\delta x_j$ represents a second-order central difference for the first derivative. The divergence-free restriction is applied through the projection

$$\frac{u_i^{n+1} - u_i^*}{\Delta t} = -\frac{\delta\phi^{n+1}}{\delta x_i}, \tag{4}$$

where ϕ follows the Poisson equation

$$(\frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta y^2} + \frac{\delta^2}{\delta z^2})\phi^{n+1} = \frac{1}{\Delta t}\frac{\delta u_i^*}{\delta x_i}, \tag{5}$$

where $\delta^2/\delta x_i^2$ represents the second-order central difference of the Laplacian operator in the *x, y,* and *z*-direction and $\delta\phi^{n+1}/\delta x_i$ is the Einstein notation for $\delta u_1^*/\delta x + \delta u_2^*/\delta y + \delta u_3^*/\delta z$. The pressure can be recovered from $p^n = \phi^n$ with a truncation error of $O(\Delta t/Re)$ [39].

To resolve the immersed boundary, the sharp-interface IB method developed by Mittal et al. [19] is adopted and the implementation has been tested extensively in the previous works [41–43].

2.3. An efficient iterative solver on the TLMR meshes

2.3.1. A parallel computation design on the TLMR meshes

Though the present TLMR does not require parallel computation, it is nevertheless most efficient to do so. The primary reason is that the mesh of the refined blocks often has a large size, especially for 3D applications, and can be stored and computed on a distributed memory system. Therefore, we parallelize the computation by sending the blocks to distributed memories and communicating between them using message passing interface (MPI). As each computing unit preserves the structured Cartesian grids, the proposed mesh refinement technique can be readily adapted from an established Cartesian grid flow solver.

The blocks of a large number of meshes can lead to coarse-grained parallelism. Besides, the computation on each Cartesian grid can be effectively multithreaded. The threading enables extra freedom to control load balance among a distributed memory system by setting a thread number for each block so the number of meshes per thread approximately equals for all blocks. Hence, the present TLMR method can benefit from hybrid parallelism to take full power of modern computer systems connected by multiple nodes with a multi-core processor. In the following discussion, we present our method for a distributed memory system.

2.3.2. Discretization on the TLMR meshes

The discretized momentum (3) and the Poisson equation (5) on the Cartesian grids can be reformulated to the following form

$$a_W\psi_W + a_E\psi_E + a_C\psi_C + a_N\psi_N + a_S\psi_S = RHS, \tag{6}$$

where $\psi_{(\cdot)}$ is the discretized value for velocity (u, v and w) or pressure (p) and $a_{(\cdot)}$ is the corresponding coefficient in the discretized equations. To simplify the discussion, we describe the descritized equations on a 2D problem as shown in Fig. 2(a), although the methodology is by no means restricted to a 2D problem.

For the 2D example, a 5-point stencil is used for the discretization of each cell, and similarly, a 7-point stencil is required for a 3D case. As mentioned earlier, data of different blocks are stored on distributed memories. The discretization scheme requires a ghost cell when performed at the boundary of each block. Similar to a domain decomposition approach in parallel computing, a layer of ghost cells are arranged at the block interface, as illustrated in Fig. 2(b). A block may have an outer ghost cell layer if it resides in a coarse block and multiple inner ghost cell layers if it contains refined blocks.

2.3.3. An iterative solver and block communications on the TLMR meshes

The discretized equation (6) can be solved using an iterative algorithm so a convergent solution can be achieved on all blocks. Some iterative methods such as Jacobi or successive over-relaxation (SOR) can be adopted, or one can use the popular Krylov subspace methods such as generalized minimal residual method (GMRES) [44] and the biconjugate gradient stabilized

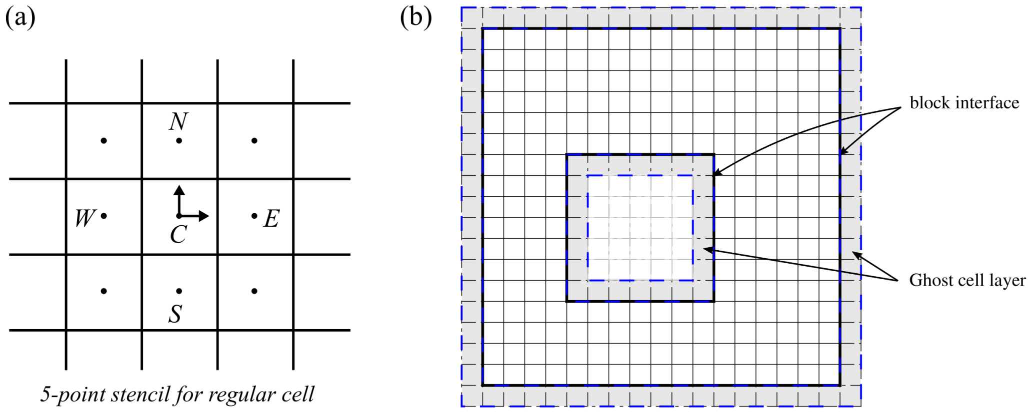

Figure 2: Illustration of discretization stencil and the ghost cell arrangement for a refined block: (a) a 5-point discretization stencil for a fluid cell and (b) the arrangement of ghost cell layers (shaded) around the block interfaces.

method (BiCGSTAB) [45]. In this study, an incomplete LU factorization method, modified strongly implicit procedure (MSIP) [46–48], is adopted for its simple implementation and fast convergence. To further improve the computation speed on a multi-core computer system, the MSIP algorithm is threaded.

To proceed with the iterative procedures on the block refinement meshes, ghost cell values need to be synchronized among the distributed memories. To communicate between blocks, the two types of block connections, as shown in the tree topology in Fig. 1(b), need to be considered. The first is the interlayer connection of two blocks between two different refinement levels, and the second is the intralayer connection of two blocks in the same refinement level.

Interlayer communication with multidimensional Lagrange interpolation

Interlayer communication synchronizes ghost cell values between the coarse and fine blocks. Data are stored at cell center, which is not coincident between coarse and fine blocks. Synchronization using the naive cell-averaged values potentially degrades the overall accuracy of the proposed mesh refinement technique. In the current approach, a multidimensional Lagrange interpolation is adopted to interpolate the cell-centered values to the right position in an interlayer communication

$$\psi(x,y,z) = \sum_{k=N_1}^{N_2} \sum_{j=M_1}^{M_2} \sum_{i=L_1}^{L_2} \psi_{ijk} r_i(x) s_j(y) t_k(z), \tag{7}$$

where $\psi(x,y,z)$ is the value to be interpolated at cooridinate (x,y,z) and ψ_{ijk} are the values of ψ at the Cartesian cells $\{[x_{L_1},\cdots,x_{L_2}]\times[y_{M_1},\cdots,y_{M_2}]\times[z_{N_1},\cdots,z_{N_2}]\}$. The one-dimensional Lagrange polynomials $r_i(x)$, $s_j(y)$, $t_k(z)$ are defined at the x, y, and z directions respectively as

$$r_i(x) = \prod_{i'\neq i,\, L_1\leq i'\leq L_2} \frac{x - x_{i'}}{x_i - x_{i'}}, \quad s_j(y) = \prod_{j'\neq j,\, M_1\leq j'\leq M_2} \frac{y - y_{j'}}{y_j - y_{j'}}, \quad t_k(z) = \prod_{k'\neq k,\, N_1\leq k'\leq N_2} \frac{z - z_{i'}}{z_k - z_{k'}},$$

with $i\in[L_1,L_2]$, $j\in[M_1,M_2]$ and $k\in[N_1,N_2]$. Fig. 3 illustrates synchronization of ghost cell values on a 2D mesh, which contains both inter- and intralayer connections, as shown in Figure 3(a). A 3×3 interpolation stencil, marked by blue square in Fig. 3(b), is used to interpolate the coarse cell values to the ghost cell in the fine block. Meanwhile, a 4×4 stencil is used to interpolate from the fine to the coarse. Likewise, for a 3D simulation, the interpolation stencils are $3{\times}3{\times}3$ and $4{\times}4{\times}4$ respectively. This interpolation strategy guarantees at least second-order accuracy in space and is crucial for the spatial accuracy of the mesh refinement technique.

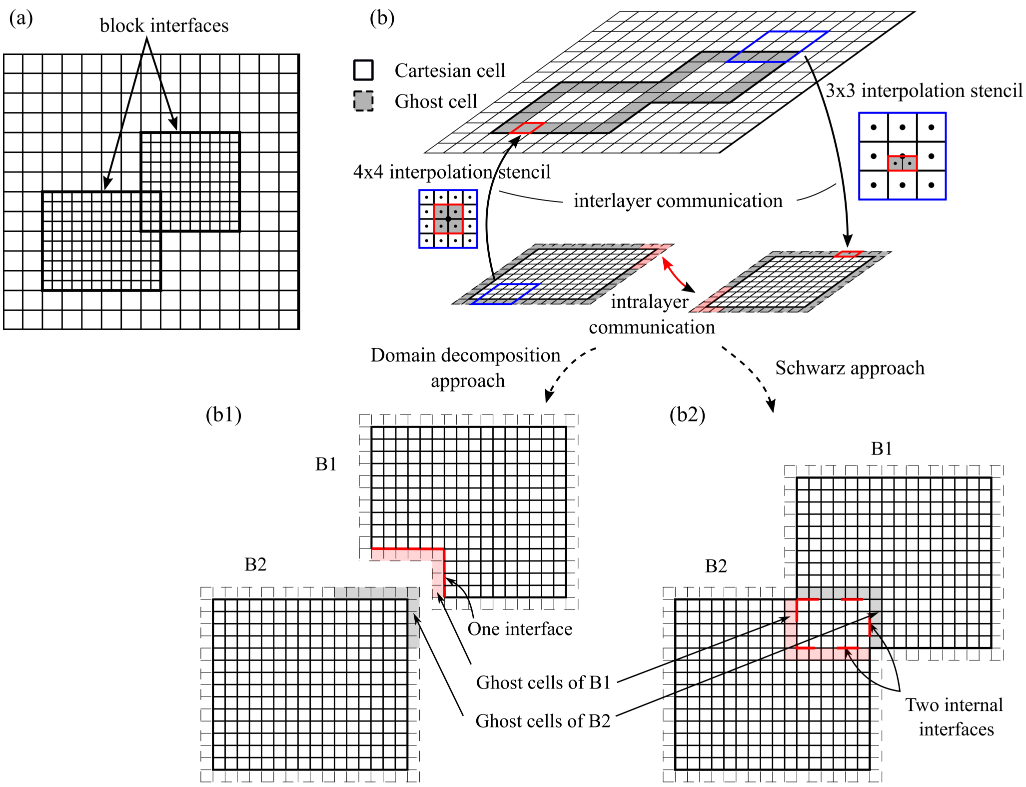

Figure 3: Schematic of information synchronization between refinement blocks: (a) a three-block mesh refinement example, (b) the interlayer and intralayer communication for ghost cells (shaded) at the block interface. For the 2D example, the interlayer communication requires to interpolate from coarse to fine cells (or vice versa), through the surrounding cells using a 3×3 (or 4×4) stencil, marked by the blue squares, to the target cells, marked by the red squares (color online), and (b1, b2) two strategies to synchronize the ghost cell values among two intraconnected blocks: (b1) synchronization across one interface and (b2) synchronization across two interfaces.

Intralayer communication using the Schwarz method for fast convergence

The intralayer-connected blocks create another situation that may influence the performance of an iterative solver and perplex the communication on the TLMR meshes. As an intralayer-connected block has finer cells and more accurate values than the coarse parent block and no interpolation is needed, it is more reasonable to synchronize the ghost-cell value from this fine block rather than the coarser parental block. In general, two strategies can be adopted in this scenario.

A straightforward strategy to communicate the ghost cell values is inspired by the domain decomposition approach in which the information is exchanged through the ghost cells around an interface between two blocks, as illustrated by Fig. 3(b1). In this approach, one block is cut at the overlap region. Another potential option results from realizing that the overlapped region, as shown in Fig. 3(b2), is internal to the connected region of the two blocks, and the value in the internal region does not affect the solution. Therefore, if the boundary value of both blocks can be updated during the iteration, the iterative algorithm may achieve faster convergence. The primitive idea has been explored by Schwarz [49] and proved to be convergent, and the potential benefits of this method for parallel computing have been realized by Lions [50]. Hence, the current approach synchronizes the ghost-cell values for each block from the intralayer-connected block using the Schwarz method as shown in Fig. 3(b), and then the iterative algorithm is performed on the whole rectangular mesh.

2.4. Efficient Poisson solver on the TLMR meshes

As mentioned earlier, the Poisson equation often converges slowly and takes a great deal of computation time when solved iteratively, such as by the iterative algorithms mentioned before. For fast convergence, a multigrid method [51, 52] is often adopted when solving this equation. The main idea of multigrid to accelerate the convergence of an iterative method is to improve the fine grid solution by a global correction obtained on a coarse grid. This is particularly useful for systems like the Poisson equations that exhibit different rates of convergence for short- and long-wavelength components, as suggested by the Fourier analysis [53].

A multigrid algorithm on block refinement meshes on a shared memory system is straightforward since the multigrid algorithm can perform the prolongation and restriction operation between coarse and fine meshes effortlessly on the multi-level refinement meshes [31]. However, care is needed on the present local refinement meshes as they are stored on distributed memories and the prolongation and restriction operation invoke communicating a large volume of data between blocks [54] and may reduce overall computation speed. Besides, as pointed out by Liu and Hu [34], the algebraic multigrid method performed on the multilayer meshes, constructed from either patch-based or octree-based [55, 56], often lacks scalability for large-scale parallel computation. In the following sections, we describe strategies for a fast Poisson solver on such meshes involving the interlayer- and intralayer-connected blocks.

2.4.1. A recursive Poisson solver on the TLMR meshes

For hierarchical coarse and fine meshes, the Poisson equation is solved recursively from the coarse block to the fine ones. That is, the Poisson equation is first solved on the coarsest block and then the finer blocks, where the latter proceeds as its boundary values are interpolated from the former. This solution process can be illustrated using the two-block problem in Fig. 4. The Poisson equation is first solved on block B_0 on all fluid cells with the given Neumann boundary conditions on the far-field and the solid boundaries. Then block B_1 is solved with its boundary value ϕ_b, synchronized from the block B_0, and the Neumann boundary condition at the solid boundaries. For computation efficiency, the boundary value ϕ_b is synchronized at every iteration instead of waiting for the convergence of its parent block. This recursive Poisson solver is similar to the level-by-level solution proposed by Guillet and Teyssier [57], who have tested and verified its efficiency on an octree-based AMR approach.

Restricting the velocity divergence $\boldsymbol{\nabla} \cdot \boldsymbol{u}^*$ from a fine block to a coarse block

To solve the Poisson equation on each block, $\boldsymbol{\nabla} \cdot \boldsymbol{u}^*$, the right-hand side of the discretized Poisson equation (5), needs to be synchronized from fine blocks to coarse ones since the intermediate velocity $\boldsymbol{u}^*$ is not stored in the refined region. Restricting $\boldsymbol{\nabla} \cdot \boldsymbol{u}^*$ from fine cells to the

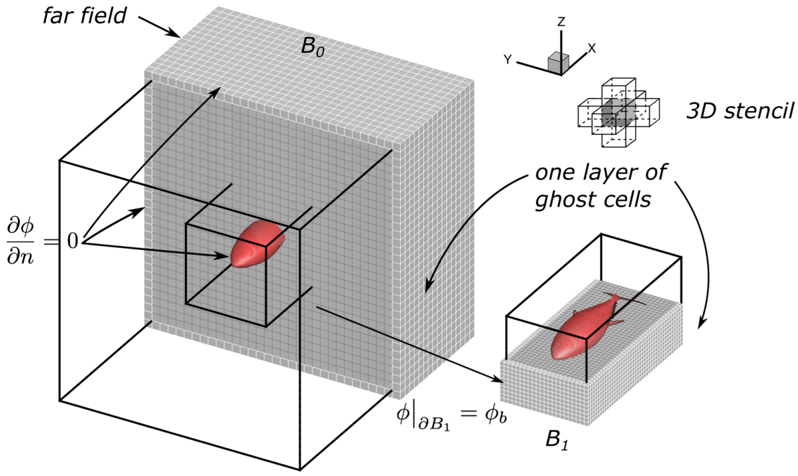

Figure 4: Schematic of solving the Poisson equation recursively from a coarse block to a fine block, of which the boundary values on the ghost cells are interpolated from the coarse block.

coarse ones on an uniform Cartesian mesh yields

$$(\nabla \cdot \boldsymbol{u}^*)^l = \frac{1}{2^N} \sum (\nabla \cdot \boldsymbol{u}^*)^{l+1} \,, \tag{8}$$

where N is the dimension of the problem and l indicates the level of refinement, with level $l+1$ mesh subdivides that of level l, as illustrated n Fig. 5. This may be illustrated with reference to the 3D example in Fig. 5(a). The divergence component $\delta u^*/\delta x$ of cell (i, j, k) at the refinement level l can be derived from the surface velocity $u^*_{i\pm1/2,j,k}$ by

$$\begin{aligned}
\left(\frac{\delta u^*}{\delta x}\right)^l_{ijk} &= \frac{u^*_{i+\frac{1}{2},j,k} - u^*_{i-\frac{1}{2},j,k}}{\Delta x} = \frac{\frac{1}{4}\sum_{k'=2k-1}^{2k}\sum_{j'=2j-1}^{2j} u^*_{2i+\frac{1}{2},j',k'} - \frac{1}{4}\sum_{k'=2k-1}^{2k}\sum_{j'=2j-1}^{2j} u^*_{2i-\frac{3}{2},j',k'}}{\Delta x} \\
&= \frac{1}{8}\frac{\sum_{k'=2k-1}^{2k}\sum_{j'=2j-1}^{2j}\sum_{i'=2i-1}^{2i}\left(u^*_{i'+\frac{1}{2},j',k'} - u^*_{i'-\frac{1}{2},j',k'}\right)}{\frac{1}{2}\Delta x} = \frac{1}{8}\sum_{k'=2k-1}^{2k}\sum_{j'=2j-1}^{2j}\sum_{i'=2i-1}^{2i}\left(\frac{\delta u^*}{\delta x}\right)^{l+1}_{i'j'k'} .
\end{aligned}$$

where cells $(i', j', k') \in [2i-1, 2i] \times [2j-1, 2j] \times [2k-1, 2k]$ at level $l+1$ are the subcells of the cell (i, j, k). Similar relations hold for $\delta v^*/\delta y$ and $\delta w^*/\delta z$, therefore yield Eqn. (8).

When solid boundaries cut through the subcells, $\boldsymbol{\nabla} \cdot \boldsymbol{u}^*$ restricts to coarse cells in a manner to preserve the cell average as indicated by Eqn. (8), with solid subcells assigned value 0. For instance, in the example in Fig. 5(b), cell $(i+1, j)$ will contain the average value of three fluid cells and one solid cell. If the coarse cell is solid, like cell $(i+2, j)$, the restricted $\boldsymbol{\nabla} \cdot \boldsymbol{u}^*$ from fine subcells will be evenly redistributed to the surrounding fluid cells to keep the conservation of $\boldsymbol{\nabla} \cdot \boldsymbol{u}^*$.

2.4.2. Parallel Schwarz method for intralayer-connected refinement blocks

When intralayer-connected blocks appear, our experience shows that the Schwarz method introduced in § 2.3.3 almost leverages the full power of a multigrid algorithm and converges much

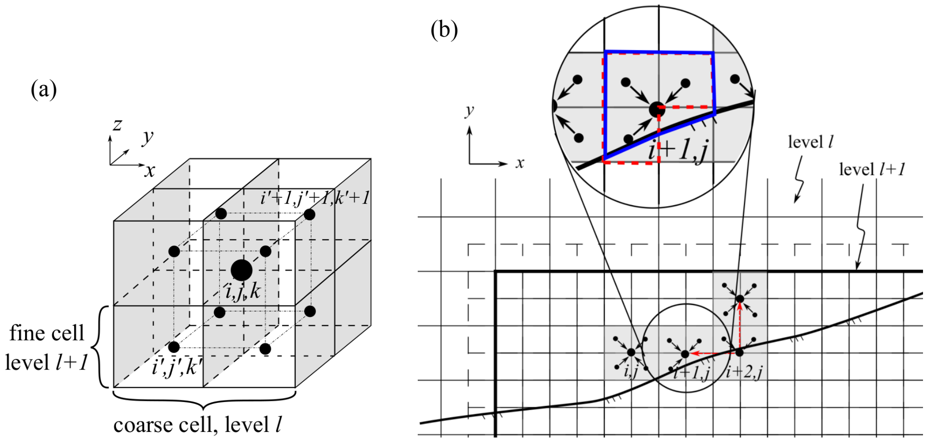

Figure 5: Restriction of the velocity divergence $\nabla \cdot \boldsymbol{u}^*$ from fine cells to a coarse cell: (a) restriction of regular 3D cells and (b) restriction of 2D cells near a solid boundary.

faster than a domain decomposition approach. The two-interfacial exchanging approach allows much efficient information exchange across the overlapped refinement blocks during the multigrid sweeps. Furthermore, the Schwarz method allows computing on each rectangular block without cutting out the overlapped region, and therefore favors a multigrid algorithm on structured mesh without the need for an algebraic multigrid method, where the latter involves extra storage and can be difficult to implement.

After the design of the recursive Poisson solver and the usage of the parallel Schwarz method, we summarize the procedures to efficiently solve the discretized Poisson equation on the TLMR meshes as follows:

Procedure 1 Procedures to solve Poisson equation on the TLMR meshes

1. Initialize the Pressure by guessed ϕ^0.
2. Compute the divergence rate $\nabla \cdot \boldsymbol{u}^*$ by Eqn. (5) on each block with values in the refined regions synchronized from fine blocks by Eqn. (8).
3. Enforce boundary conditions around the computational domain.
4. Continue following iterations.
   (a) For each refined block, synchronize values ghost cell layer from the coarse parental block. If an intralayer connection exists, replace values of the ghost cell layer of the connected region from the connected block.
   (b) Enforce boundary conditions around the immersed solid via an IBM method.
   (c) For each refinement block, solve the Poisson equation on the Cartesian grids using a multigrid method and accelerate the computation with multithreading if needed.
   (d) Check the convergence of Eqn. (5): if yes, exit iteration; otherwise, return to step 4a.

3. Results and discussion

In this section, we assess the accuracy and efficiency of the present TLMR method and demonstrate its application to bio-inspired flow simulations, especially the simulation of groups of fish swimming together. Firstly, a convergence study is performed using two prototypical flow problems to verify its spatial and temporal discretization accuracy. Secondly, two canonical flow problems with stationary or moving boundaries are simulated and compared to the references. Finally, we simulate flow with highly complex, non-canonical geometries to showcase the capabilities of the solver for complex bio-inspired flow. The computations were performed on a supercomputer that has up to 575 nodes with over 20476 cores. For performance evaluation, nodes used for computation contain dual Intel Xeon E5-2680v3 twelve-core processors with a CPU frequency of 2.5GHz.

3.1. Convergence study of the numerical solver on the TLMR meshes

3.1.1. The Taylor-Green Vortex problem

To demonstrate the spatial and temporal discretization accuracy of our algorithm on block refinement meshes, we first consider the Taylor Green vortex flow [58], which is an unsteady flow with decaying or growing vorticity on a periodic domain. It usually starts with an initial condition

$$\begin{aligned} u &= A\cos ax \sin by \sin cz, \\ v &= B\sin ax \cos by \sin cz, \\ w &= C\sin ax \sin by \cos cz, \end{aligned}$$

where $Aa + Bb + Cc = 0$ because of the continuity equation. Exact solutions exist on a 2D domain. Within a $2\pi \times 2\pi$ periodic domain, the solution can be written as

$$\begin{aligned} u &= \cos x \sin y F(t), \\ v &= -\sin x \cos y F(t), \\ p &= -\frac{\rho}{4}(\cos 2x + \cos 2y)F^2(t), \end{aligned} \tag{9}$$

where $A = -B = a = b = 1$ and $F(t) = e^{-2\nu t}$ is a decaying function. ν is the kinematic viscosity of the fluid and is related to the Reynolds number by $\nu = UL/Re$, where U and L being the reference velocity and length. For simplicity, the Reynolds number for the investigation is chosen 50.

We first demonstrate that the multi-dimensional Lagrange interpolation is critical for the spatial accuracy. We discretize the $2\pi \times 2\pi$ domain and a refined $\pi \times \pi$ region at the center by a 32×32 mesh. Fig. 6(a) and Fig. 6(b) show the contour plots of streamwise velocity (u) and pressure (p). Both the velocity and pressure match the exact solution well if the interlayer communication employs the multi-dimensional Lagrange interpolation, as shown in Fig. 3. We also tried a simple bilinear interpolation, which yields much bigger error in pressure, as shown in Fig. 6(b). Even worse, the error $||\boldsymbol{e}_u||_{2,BL}$ shows that the velocity is only first-order accurate in space with the bilinear interpolation. The normalized global error for velocity is defined by

$$||\boldsymbol{e}_{\boldsymbol{u}}||_2 = \frac{||\boldsymbol{u} - \boldsymbol{u}_e||_2}{||\boldsymbol{u}_e||_2} = \sqrt{\left(\frac{\int_\Omega ||\boldsymbol{u} - \boldsymbol{u}_e||^2 dA/A}{\int_\Omega ||\boldsymbol{u}_e||^2 dA/A}\right)},$$

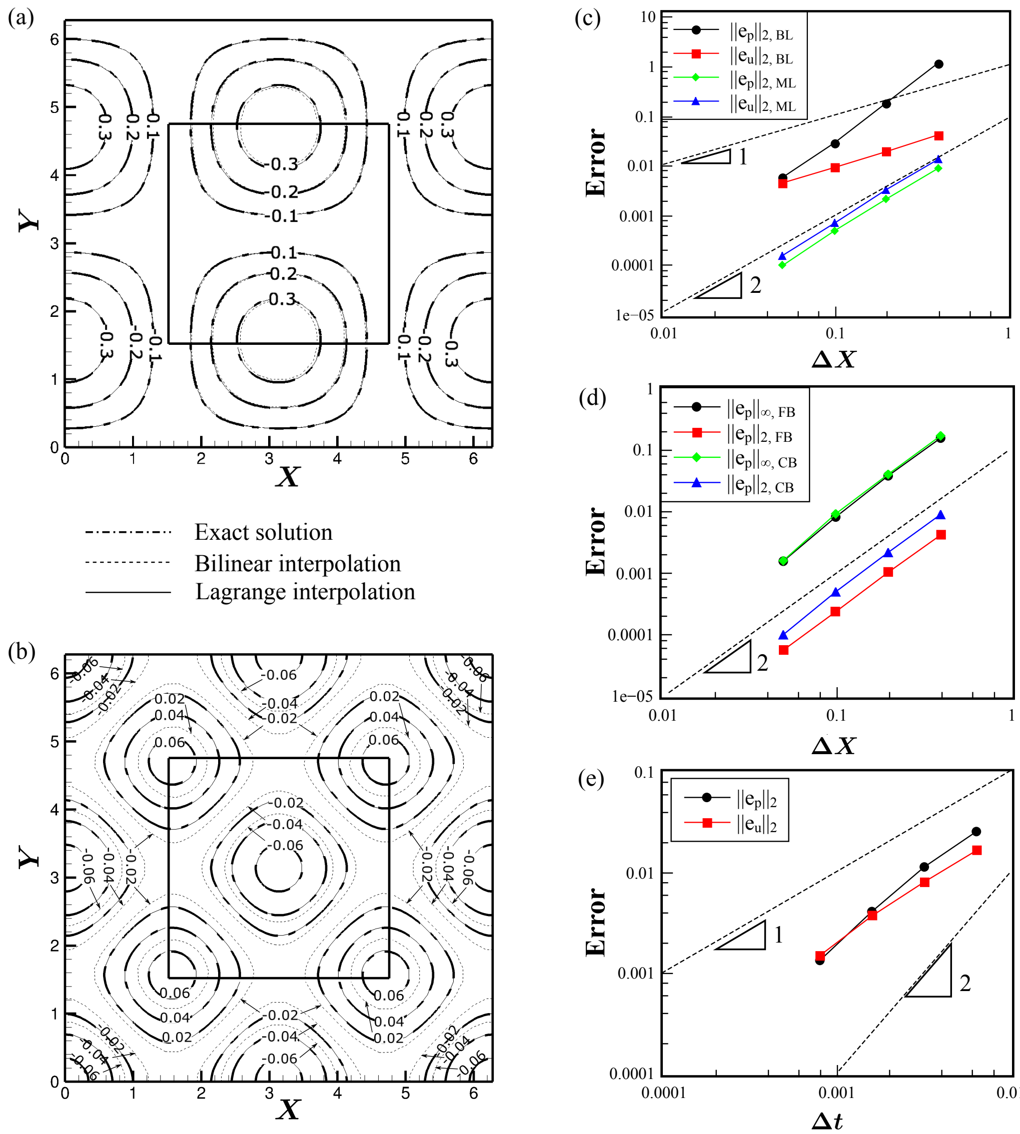

Figure 6: Convergence study of the TLMR-IBM flow solver using the 2D Taylor-Green vortex flow problem: (a) and (b) contour plot of the x-component velocity u and the pressure p using bilinear interpolation or multi-dimensional Lagrange interpolation for the interlayer communication, (c) error when using the bilinear interpolation (BL) and the multi-dimensional Lagrange interpolation (ML) for the interlayer communication, (d) grid convergence study of the pressure on both coarse block (CB) and fine block (FB), and (e) temporal convergence study for the velocity u and pressure p.

and a normalized local error is defined by

$$\|\boldsymbol{e}_{\boldsymbol{u}}\|_\infty = \frac{\|\boldsymbol{u}-\boldsymbol{u}_e\|_\infty}{\|\boldsymbol{u}_e\|_2} = \frac{\max|\boldsymbol{u}-\boldsymbol{u}_e|}{\sqrt{\int_\Omega \|\boldsymbol{u}_e\|^2 dA/A}},$$

where $\boldsymbol{u}_e$ are the exact solution (9), Ω is the flow domain, and A is the area of the domain. Error for the pressure can be similarly defined. We compute the errors on different grids 32×32, 64×64, 128×128 and 256×256, so the finest grid resolutions are $(\pi/32)\times(\pi/32)$, $(\pi/64)\times(\pi/64)$, $(\pi/128)\times(\pi/128)$ and $(\pi/256)\times(\pi/256)$, respectively. Figure 6(c) shows that using the multi-dimensional Lagrangian interpolation for ghost cell values, the solver can reach a global second-order accuracy for both velocity and pressure. For the pressure, as shown in Fig. 6(d), a global and local second-order accuracy in space for both the find block (FB) and the coarse block (CB) is observed via the error $||e_p||_{2/\infty,FB/CB}$. We further repeat the the simulation on the 64×64 with different time steps Δt, $\Delta t/2$, $\Delta t/4$, $\Delta t/8$, where $\Delta t = 0.00633$, and the error in Fig. 6(e) shows that the temporal accuracy of solver is between the first- and second-order.

3.1.2. Flow past a fixed boundary

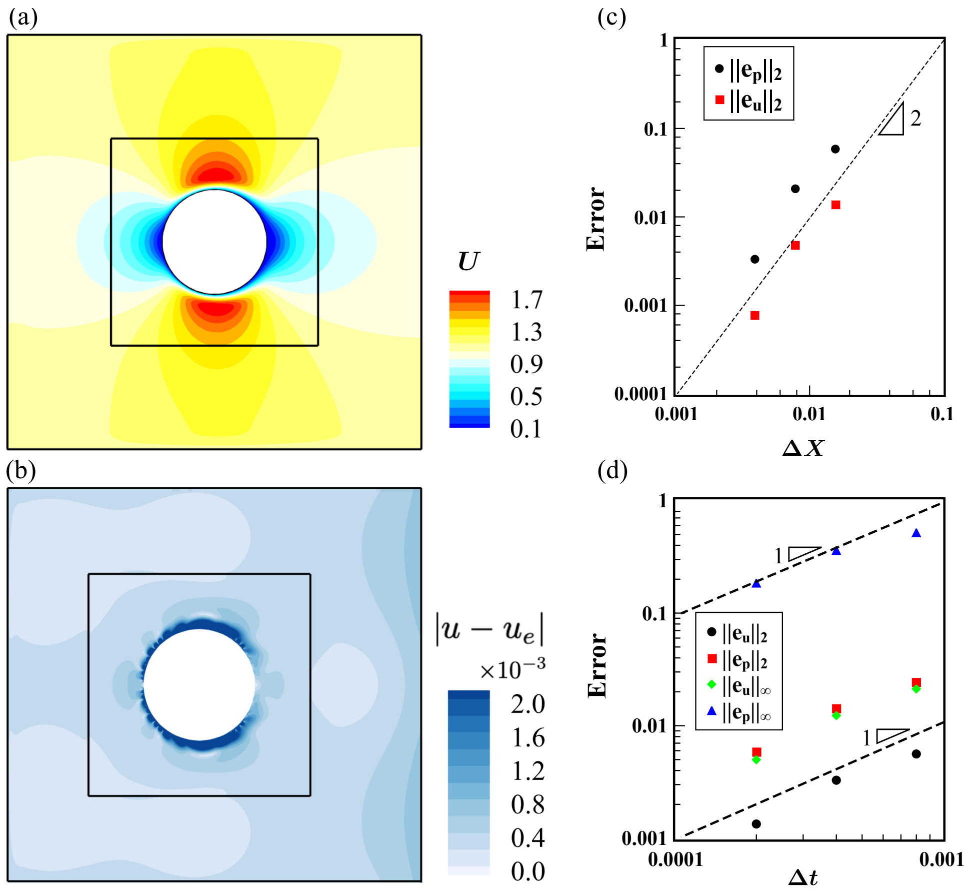

Figure 7: Flow past stationary cylinder problem for the convergence study of the TLMR-IBM flow solver when solid boundary immersed: (a) contour plot the x-component velocity u on a 256×256 uniform mesh at $t = 0.2D/U_\infty$, (b) error of velocity $|\boldsymbol{u}-\boldsymbol{u}_e|$ at $t = 0.2D/U_\infty$, where $\boldsymbol{u}_e$ is the reference values computed on a uniform 2048×2048 mesh, and (c) and (d) the spatial and temporal convergence of the TLMR-IBM solver respectively.

To investigate the accuracy of the developed algorithm with solid boundaries immersed, we consider the flow past a fixed circular cylinder. The Reynolds number $Re = U_\infty D/\nu$ is chosen 100, where D is the diameter of the cylinder. For this test, we adopt a uniform Cartesian mesh on a $4D \times 4D$ computational domain, and a $2D \times 2D$ refined block at the center. For reference, we use the numerical results from the in-house numerical solver [19] on a high-resolution grid (2048×2048) as the reference value, since the solver is well benchmarked. We choose a time step $0.0001D/U_\infty$ and three sets of meshes, 128×128, 256×256 and 512×512 for both coarse and fine blocks. The results are compared with the reference solution at the 2000^{th} step.

Figure 7(a) shows the streamwise velocity on a 256×256 mesh. The error of the velocity field, as shown in Fig. 7(b), indicates that the maximum error appears around the solid boundary. Mesh refinement around a solid body therefore can improve the accuracy at the boundary. The error $||\boldsymbol{e_u}||_2$ and $||\boldsymbol{e}_p||_2$ in Fig 7(c) shows that velocity $\boldsymbol{u}$ and pressure p both achieve a global second-order accuracy in space. The error study in Fig 7(d) shows that the solver is approximately first-order accurate in time for both velocity and pressure. This is consistent to the conclusion that the truncation error of $p = \phi$ is $O(\Delta t/Re)$ [39], and the error may be less significant when Reynolds number is high.

3.2. Benchmark cases for simple flows

In this section, we further benchmark the developed TLMR-IBM flow solver using two classical flows, with either stationary or moving boundaries, and compare the results with the references.

3.2.1. Two-dimensional flow past a fixed circular cylinder

In many bio-inspired flow problems, the fluid is usually blocked by the body and the circulation around the body may create two rolls of vortices after the body. This phenomenon is similar to the classical problem of flow past a fixed cylinder, which is studied here to mimic the flow past the body of various swimmers. For simplicity, we only consider 2D cases with Reynolds numbers in a range of $10^2 \sim 10^4$, as it is noticed that the Reynolds numbers for bio-inspired flow usually reside in the range of $10^2 \sim 10^6$ for insects, birds, and marine animals [59–61]. And among them, the flow is dominated by the moving boundaries and the shear layer instabilities typically when $Re < 10^5$.

As shown in Fig. 8(a), the size of the computational domain is $38D \times 18D$, the same as the study of Singh and Mittal [62]. $D = 1$ is the diameter of the cylinder. Dirichlet boundary condition is set at the left, top, and bottom boundaries, while Neumann condition is applied at the outlet. The domain is discretized using the stretched Cartesian grids around a uniform region of size $12D \times 8D$ with the resolution of 0.016×0.016 around the cylinder, which is located $(10, 9)$. The Cartesian grids provide the background meshes for the refined blocks. To simulate the flow under various Reynolds numbers, up to 4 layers of refinement meshes are adopted, as illustrated in Fig. 8(b). To simulate the flow under different Reynolds numbers, the number of blocks used and the corresponding resolution are listed in Table 1.

The computed drag forces on the cylinder are shown in Fig. 8(c). At low Reynolds numbers, 100 and 200, the drag forces are close to the experimental results of Wieselsberger (taken from [62]) and Tritton [63]. Since flow past a cylinder develops 3D structures at high Reynolds numbers, the 2D numerical simulation does not fit well with experimental results when the Reynolds number is higher than 200. Therefore, for a higher Reynolds number, it is reasonable to compare with other 2D numerical studies. For instance, in the range of $100 \sim 1000$, our numerical results are close to Henderson [64], who computed the unsteady flow using a spectral element method. The drag force also matches that obtained by Beaudan and Moin [65],

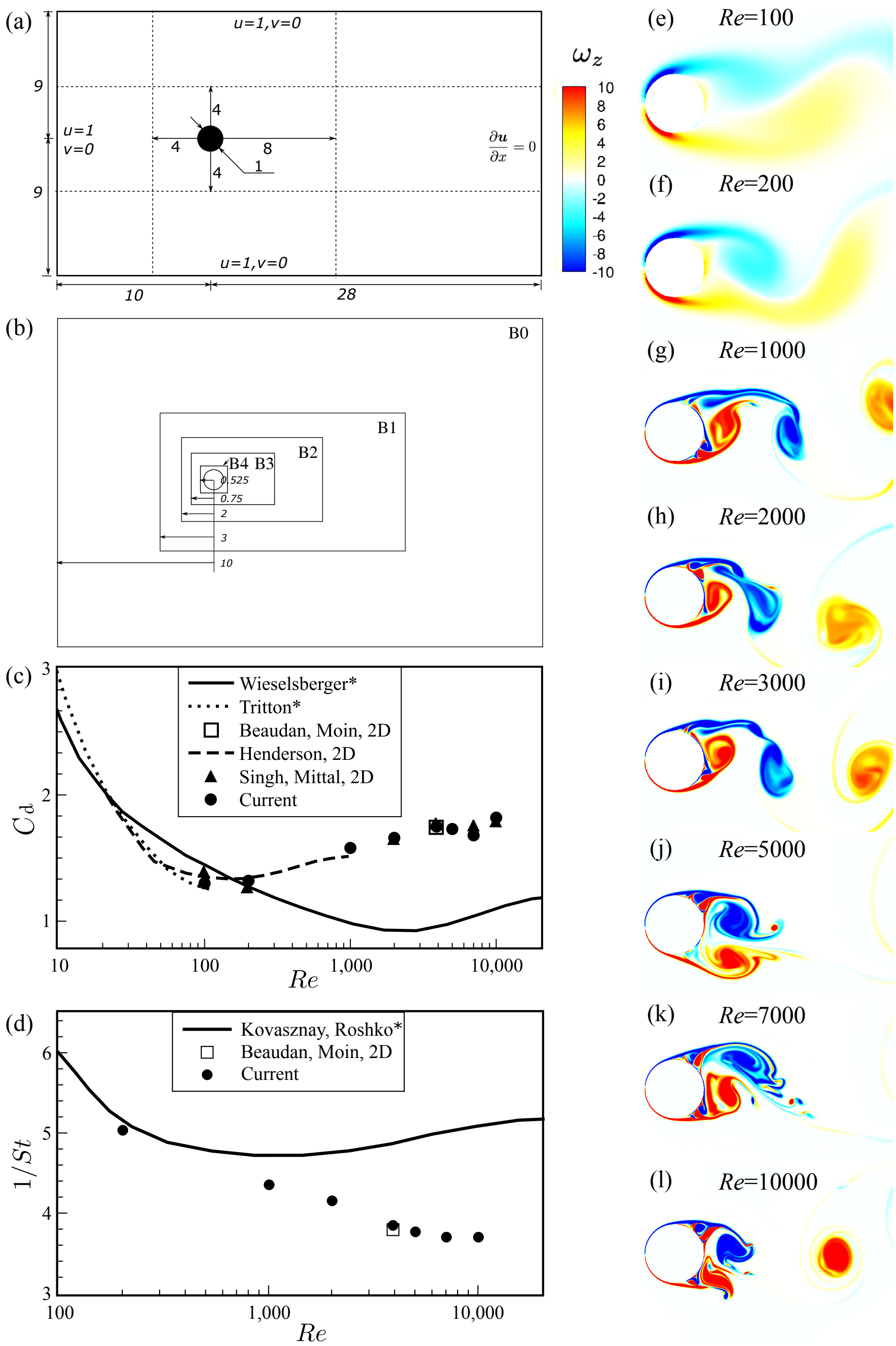

Figure 8: A 2D simulation of flow past stationary cylinder under various Reynolds numbers: (a) set up of the 2D simulation, (b) refinement blocks configuration, (c) and (d) the drag coefficient and Strouhal number for different Reynolds numbers and (e) ~ (l) the corresponding wake structures. '*' in (c) and (d) indicates experimental data.

Table 1: Refinement blocks for 2D cylinder simulation

Block	Block size	Grid resolution ($\Delta = \Delta_x = \Delta_y$)	Number of meshes ($\times 10^6$)	Reynolds number
0	$38D \times 18D$	0.016	0.39	100, 200
1	$10D \times 4D$	0.008	0.98	-
2	$5D \times 2D$	0.004	1.64	1000, 2000, 3900
3	$3D \times 1.5D$	0.002	2.82	5000, 7000
4	$1.05D \times 1.05D$	0.001	4.15	10000

who performed a numerical study for the 2D flow at $Re = 3900$. For higher Reynolds number ($10^3 \leq Re \leq 10^4$) our results match that of Singh and Mittal [62], who numerical studied the 2D shear layer instability using a finite element approach with deliberately fine layers of meshes surrounding the cylinder to resolve flow in the boundary layer.

The shedding period of the primary vortex (measured by the reciprocal of Strouhal number) is plotted in Fig. 8(d). At low Reynolds ($100 \sim 200$), the numerical results match the experimental data by Kovasznay and Roshko (extracted from [66]). At $Re = 3900$, the computed frequency matches those computed by Beaudan and Moin [65]. For higher Reynolds numbers, the numerically captured shedding period gently decreases with increased Re.

The instantaneous vortices after the cylinder provide visual information about the primary and shear layer instabilities. At a low Reynolds number ($100 \sim 200$), there are only two organized vortices after the cylinder. After $Re > 1000$, small shear layer instability develops but does not influence the far wake. Further increasing Re, shear layer instability becomes significant at the top and bottom of the cylinder. At $Re = 10^4$, the primary vortices are significantly reduced and the shear layer vortices start to dominate the near wake region.

Besides the aforementioned coincidence of statistical comparison with the literature, some properties in the near wake region can illustrate the value of the TLMR method in resolving flow inside the boundary layer. Figure 9(a) shows the base suction pressure coefficient C_{pb} which is given by

$$C_{pb} = \frac{\bar{p}_{pb}}{\frac{1}{2}\rho U_\infty^2},$$

where $\bar{p}_{pb}$ is the time-averaged pressure at the rear of the cylinder ($\theta = 180^\circ$ as referenced in Fig. 10(a)). At a low Reynolds number, $Re = 100$ to be specific, the computed base pressure is coincident with the experimental results of Williamson and Roshko [67]. When $Re = 200$, a 3D flow structure yielded from secondary flow instability will cause a sudden drop of pressure [68]. Without modeling the 3D structures, our simulation only matches other 2D simulations. For instance, when $Re = 3900$, our base pressure matches that of Baudan and Moin's results. In a wide range of $Re > 2000$, our simulations are similar to that of Singh and Mittal, but with a slight difference. This might be because our finest Cartesian mesh is still relatively coarse compared to that of Singh and Mittal, who adopted a boundary layer thickness around 10^{-5} compared to 10^{-3} in the current study. We further compare the pressure distribution along the surface of the cylinder to that of Singh and Mittal for the verification of the refinement technique. Three different Reynolds numbers, 100, 2000, and 10000, are compared here. Our simulations show good coincidence when $\theta < 80^\circ$, and slight discrepancies in the separation region.

Figure 10 plots the velocity profiles of the laminar boundary layer at the Reynolds number of

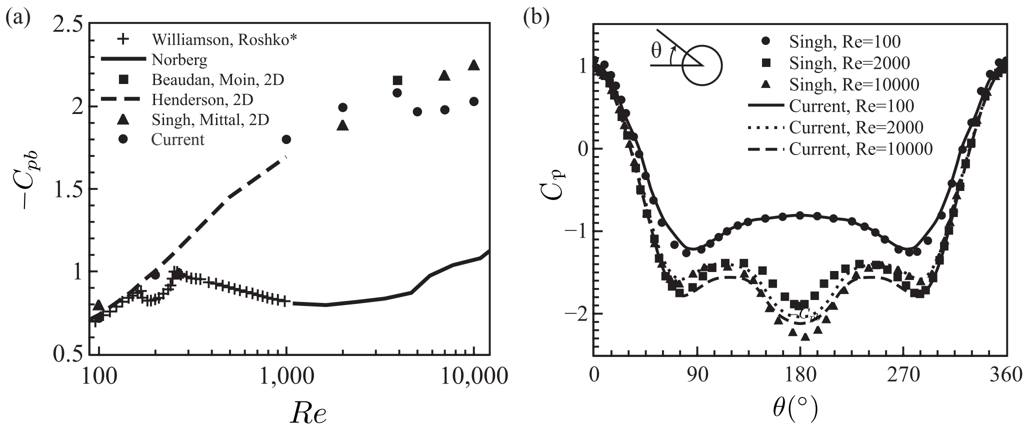

Figure 9: Benchmark of 2D flow past a stationary cylinder: (a) base suction pressure coefficient $-C_{pb}$ and (b) averaged pressure distribution over the cylinder. '*' in (a) indicates experimental data.

10^4. The velocity is plotted in the local cooridinate, where the shear velocity is tangential to the wall and varies along the wall normal direction, as shown in the top-right corner of Fig. 10(a). The shear velocity and wall norm distance is normalized by

$$u^+ = \frac{u}{u_\tau}, \quad y^+ = \frac{y u_\tau}{\mu},$$

where u_τ is the shear velocity at the wall computed by $u_\tau = \sqrt{\tau_w/\rho}$ and τ_w is the wall shear stress computed by $\tau_w = \mu \partial u/\partial y|_{y=0}$. In the current simulation, the y^+ of the cells is around 4, providing sufficient resolution for the viscous boundary layer. The flow separates from the cylinder after $\theta \geq 90°$, and therefore is not discussed here.

In the laminar boundary layer that attaches to the cylinder surface, the velocity profile along the wall-normal direction at different locations of θ is plotted in Fig. 10(a). The laminar velocity profile deviates from curve $u^+ = y^+$, which is valid for the viscous sublayer in a turbulent boundary layer. For comparison, we include the boundary profiles derived by Hsu (see appendix F in Ref. [69]), who solved the boundary-layer equations using the wall condition and the potential flow profile as the outer boundary condition for the compressible flow. Both approaches yield similar results when the distance in the y-direction is small ($y^+ < 10$). Since the potential flow has a larger velocity than the real viscous flow, the deviation becomes large when both θ and y become large.

On the other hand, along the wall-tangential direction, due to the curvature of the cylinder, the shear velocity can be higher than the income flow and often reaches a maximum velocity u_{max} at a height of y_{max}. For the inviscid flow, the development of u_{max} with respect to different θ is given by a simple sinusoid relation

$$u(\theta) = U_0 \sin(\theta), \tag{10}$$

where U_0 is the incoming flow velocity. In the real flow, the velocity is retarded by the viscosity and can not reach such a high value. As the boundary layer develops along with the circular cylinder, as shown in Fig. 10(a), the thickness y_{max} and the maximum velocity u_{max} increase and the curve $u^+ = y^+$ approximately intercepts the maximum velocity from the velocity profiles at different angles of θ. The thickness y_{max} and maximum velocity u_{max} are further plotted in Fig. 10(b). When the boundary layer attaches to the surface of the cylinder, second or third-order polynomials fit well the data. Applying the boundary condition that the thickness y_{max} is symmetric and u_{max} antisymmetric at $\theta = 0$, the data are fitted by the lines

$$\begin{aligned} \sqrt{Re}\, y_{max} &= 1.47 \times 10^{-4}\theta^2 + 1.38, \\ u_{max} &= -1.77 \times 10^{-6}\theta^3 + 0.0326\theta, \end{aligned}$$

where the lines are plotted aside the data points using dashed lines. From the fitted curves, the thinnest thickness is around 0.0138, and the maximum velocity in the boundary layer is around 1.7 and is below the potential flow. Our numerical results reflect this trend well. Furthermore, our numerical results are also close to the experimental results of Hiemenz at a Reynolds number of 1.9×10^4 (extracted from Ref. [70], Chapter 8.3.3). In the region of favorable pressure gradient ($\theta < 70°$), the fitted curves of u/U_0 for numerical and experimental data are close despite slightly different Reynolds numbers. In the adverse pressure gradient region, the experiments of Hiemenz clearly show that the potential velocity is difficult to recover at a higher Reynolds number. Our numerical simulation at $Re = 10^4$ experiences less velocity loss compared to that of Hiemenz, as shown in Fig. 10(b).

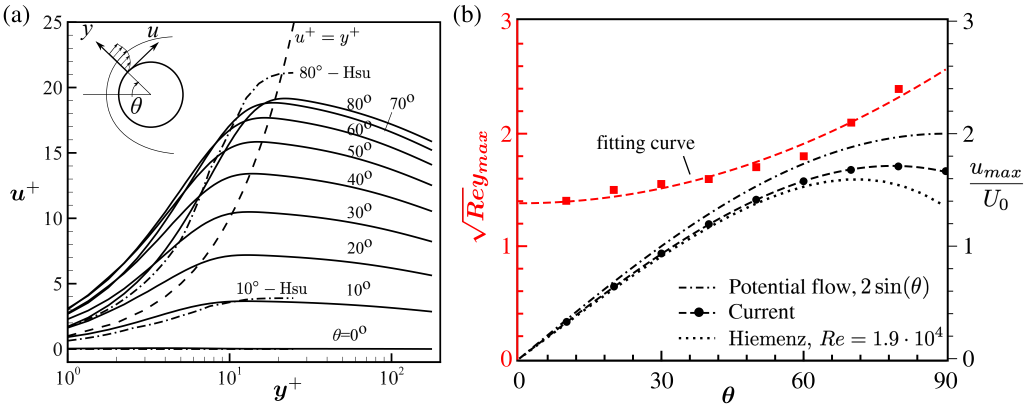

Figure 10: The velocity profile in the shear layer of a 2D cylinder at $Re = 10^4$: (a) the shear velocity profile and (b) the maximum velocity u_{max} and thickness y_{max} at different location of θ.

3.2.2. Three-dimensional flow past a pitching panel

Another important bio-inspired flow is the flow stirred by a thin revolving panel, like a fish fin or an insect wing. Inspired by this, we study a model problem with a revolving trapezoidal panel. For a fair comparison, the panel geometry and kinematics are the same as the experiments of King et al. [71].

The geometry of the panel is shown in Fig. 11(a), and the associated parameters are listed in Table 2. c is the cord length of the panel and is chosen as the reference length. All dimensions are scaled respectively. The panel pitches around the leading edge following a sine wave at a frequency of $f = 1\,Hz$ and a peak-to-peak amplitude of 15°. The Strouhal number, St, and

Reynolds number, Re, are defined as $St = fA/U$ and $Re = Uc/\nu$, respectively, where A denotes the peak-to-peak trailing edge amplitude, U is the incoming flow velocity, and ν indicates the kinematic viscosity of the fluid. Three Strouhal numbers, 0.27, 0.37, and 0.46, from the experiments, are chosen for the verification of the TLMR-IBM solver. According to the experiments, the variation of the Strouhal number is accomplished by changing the incoming flow velocity, and the corresponding Reynolds numbers are Re = 10200, 7400, and 5800, respectively.

Table 2: Summary of the non-dimensional geometry parameters in the simulation

Chord length, c	Span, b	Thickness, δ	Sweep angle	Aspect ratio, AR
1.0	2.51	0.016	45°	4.17

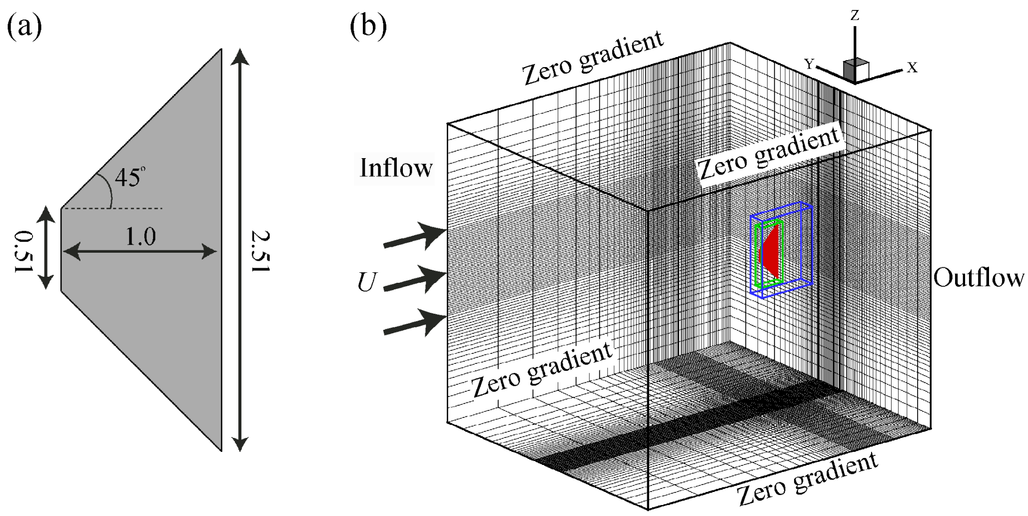

Figure 11: Schematic of panel simulation: (a) panel geometry and (b) setup of the 3D simulation.

The configuration for the numerical simulation, including the block refinement meshes and the boundary conditions are displayed in Fig. 11(b). The domain size is $16.0 \times 14.0 \times 14.0$ with total grid nodes around 5.68 million ($176 \times 144 \times 224$). To resolve the flow structures at a high Reynolds number, two layers of refined meshes are employed, so the resolution is 0.0052 around the thin plate. The total number of meshes is around 15.4 million. In contrast, the grid number can easily go up to 1 billion ($15.4M \times 8 \times 8$) without the TLMR method. The details of the refinement blocks are summarized in Table 3. A constant inflow velocity boundary condition is assigned at the left-hand boundary, and an outflow boundary is imposed at the right-hand boundary. The zero-gradient boundary condition is set at all other lateral boundaries. For the pressure condition, a homogeneous Neumann boundary is applied at all boundaries.

Figure 12 presents the vortex wake structures under three Strouhal number at two different time instance, $t = 0$ and $t = 0.25T$. Figure 12(a1-f1) are the experimentally observed wakes by King et al. [71] (courtesy of King et al.) using isosurfaces of Q-criterion [72]. Figure 12(a2-f2) shows the numerically observed wakes, which are visualized at a value of 1% of Q_{max}, and Q isosurfaces are colored by the value of the spanwise vorticity ω_z to be consistent with the ex-

Table 3: Summary of refinement blocks for a 3D pitching panel simulation

Block	Block size	Grid resolution ($\Delta = \Delta_x = \Delta_y = \Delta_z$)	Number of meshes ($\times 10^6$)
0	$16.0 \times 14.0 \times 14.0$	0.0208	5.68
1	$2.8 \times 0.8 \times 3.6$	0.0104	4.13
2	$1.18 \times 0.4 \times 2.8$	0.0052	5.63

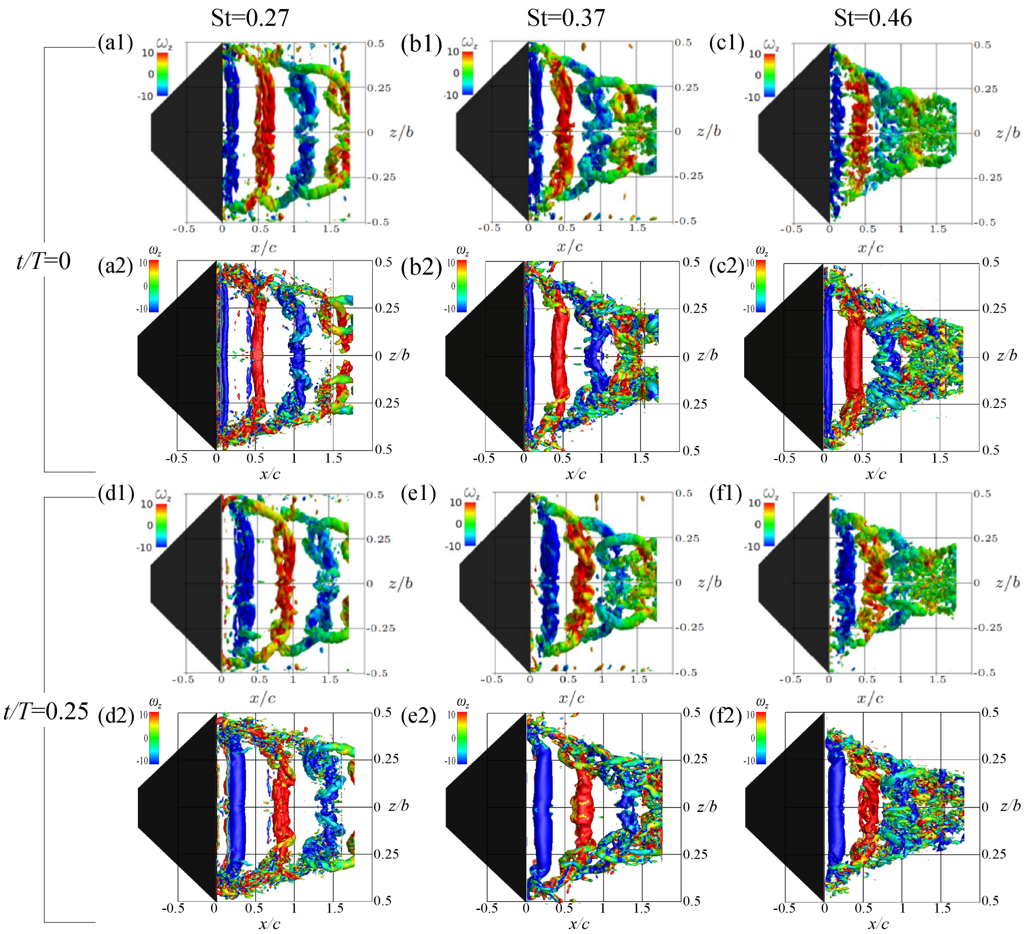

Figure 12: Snapshots of wake structures of the pitching panel: (a1, a2, d1, d2) St = 0.27, (b1, b2, e1, e2) St = 0.37 and (c1, c2, f1, f2) St = 0.46 at t = 0 (a1-c1, a2-c2) and t = 0.25 (d1-f1, d2-f2) respectively. The figures (a1-f1) are from experiments results [71] (reprinted with permission from King et al.), and (a2-f2) are from current simulations.

periments. The plot shows that the numerical simulation captures the main flow features of the unsteady flow observed in the experiments. For example, the spanwise vortices are shed from the trailing edge of the panel alternatively and form the reverse Kármán vortex street. The vortex street shrinks in the spanwise direction and gradually becomes disorganized as it convects downstream. At the same time, tip vortices are generated at the ends of the panel, connecting the neighboring spanwise vortices. Furthermore, the numerical simulations correctly reveal the dominant role of St in the development of wake structures. With increased St, the wakes are compressed heavier in the spanwise direction and the onset of wake breakdown moves upstream. For instance, at $St = 0.27$, the Q isosurface exhibits between $z/b \approx \pm 0.45$ at $x/c \approx 0.5$, and the wake breaks at $x/c \approx 1.75$ near the midspan plane (Fig. 12(a2)); while at a higher St of 0.46, the Q isosurface extents from $z/b \approx \pm 0.375$ at $x/c \approx 0.5$. In addition, the vortex tube at $x/c \approx 0.75$ become twisted and weak, and the wake breakdown begins at $x/c \approx 1.2$, where the ω_z is around zero near the midspan plane, as shown in Fig. 12(c2). The well-captured wake structures prove that the block-based mesh refinement technique could provide sufficient resolution to three-dimensional high Reynolds number problems. In addition, we present the time variation of the force coefficients for the three pitching panels in Fig. 13 for future reference. The force coefficients are defined as $C_T = -F_X/(0.5\rho U^2 S)$ and $C_Y = F_Y/(0.5\rho U^2 S)$, where S is the area of the panel. For all cases, two peaks characterize the thrust force (C_T) in one cycle, while the lateral force (C_Y) oscillates with one peak per stroke.

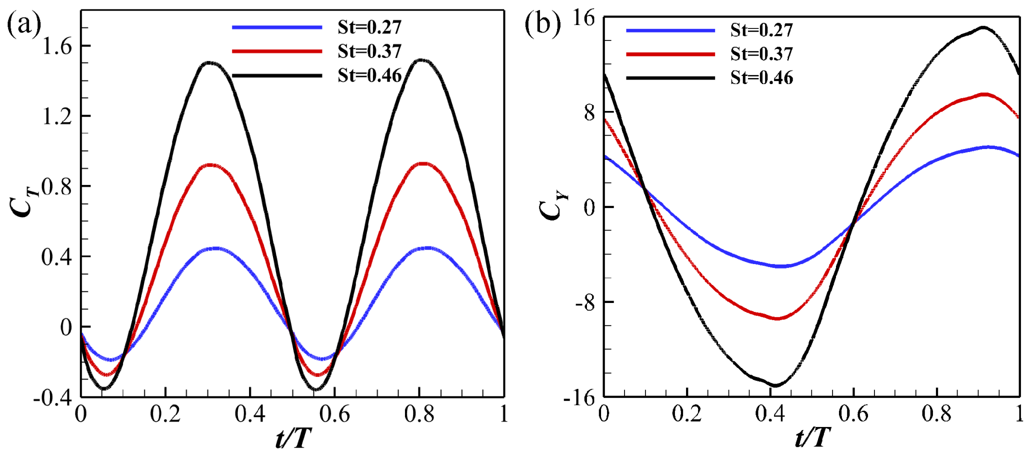

Figure 13: Time history of force coefficients for the pitching panel: (a) coefficient of thrust C_T and (b) coefficient of lateral force C_Y

3.3. The versatility and efficiency of the TLMR-IBM solver for bio-inspired flow simulations

In the following section, we will demonstrate how the TLMR-IBM solver performs when multiple objects are immersed in the flow, like the flow around and within schools of fish. Both 2D and 3D cases with moving boundaries will be presented in the following studies.

3.3.1. Simulation of the two-dimensional fish school swimming problems

When fishes swim in a school, strong nonlinear body-body interactions can improve their thrust and propulsive efficiency as revealed in a recent numerical study by Pan and Dong [73]. Such a school swimming case is an excellent test case to demonstrate the efficiency of the present TLMR method.

Figure 14(a) shows a diamond-shaped fish school formation, where L is the body length and U_∞ is the swimming speed. The S is the streamwise distance between the leading fish 1 and the trailing fish 4. The lateral spacing D is the spacing between the centers of fish 2 and fish 3. The Reynolds number for the simulation is 1000 based on the incoming flow velocity and body length. The undulating motion of fish is usually modeled by a traveling wave equation [73]. Two typical diamond-shaped schools, sparse (CASE I) and dense (CASE II), are illustrated in Fig. 14(a) and Fig. 14(b). In the sparse school, the lateral distance D is $1L$, and $0.5L$ is in the dense school. The streamwise spacing S is kept constant $0.4L$. The computation domain is of size $30L \times 20L$. With a two-layer mesh refinement, the resolution around the fishes is 0.0022×0.0022. For easier illustration, a window of $4.5L \times 3L$ around the school is plotted and the mesh is coarsened by 4 times in each direction.

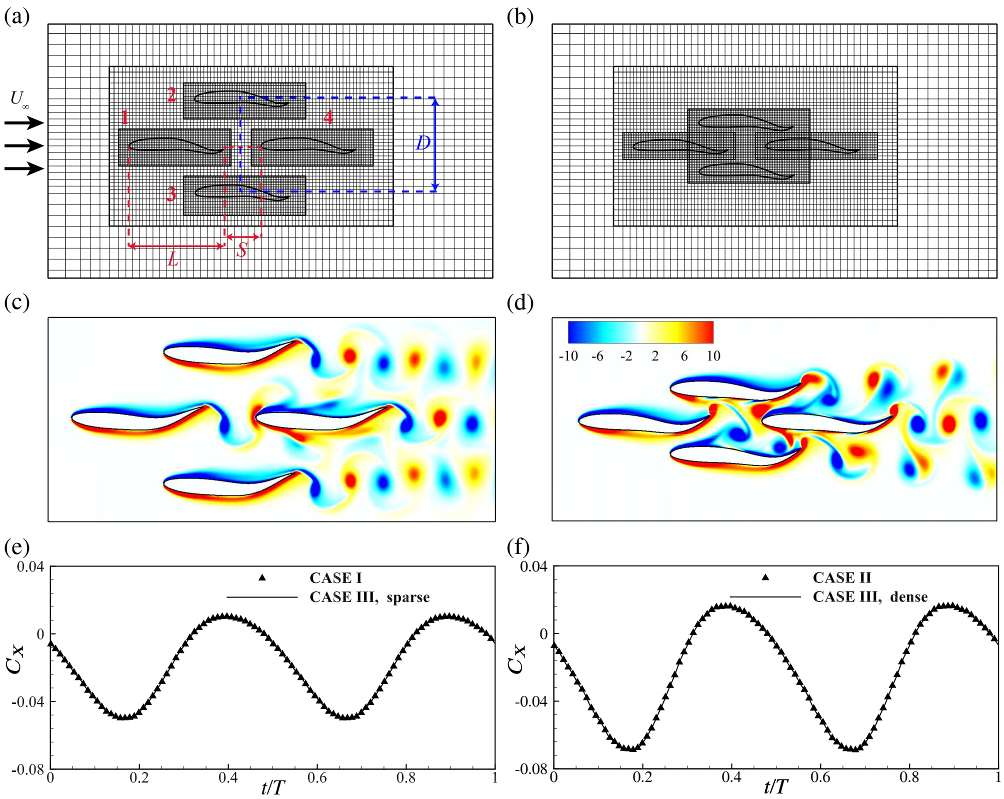

Figure 14: Numerical study of sparse and dense schools: (a) and (b) schematic of a coarse (CASE I) and a dense (CASE II) diamond-shaped fish school with definitions of quantities describing its spatial arrangement in (a), mesh are coarsen in each direction by 4 times for easier illustration, (c) the 3S vortex streets in the sparse diamond school, (d) two 2P and one 2S vortex streets in the narrow diamond school, and (e, f) the streamwise force C_x of fish 1 in one undulating motion cycle for the sparse and dense schools respectively.

The simulations produce similar vortex patterns as the reference [73]. In the sparse school, a 2S vortex street is shed from each row of fishes, as shown by Fig. 14(c). In the dense school, as shown in Fig. 14(d), the top and bottom rows of fishes shed the 2P vortex streets while the middle row sheds 2S vortex streets. The streamwise force C_X of fish 1 in the school is plotted in Fig 14(e) and 14(f). For comparison, the same school swimming problems are simulated in a single block of Cartesian grids, which are obtained by subdividing the base block in Fig. 14(a) in each direction by a factor of 4. As shown by the plots, the forces computed using the present mesh refinement technique are identical to the reference cases, which are denoted as CASE III.

Computational efficiency of the 2D fish school problems

To compare computational efficiency, Table 4 lists the runtime information of the above two simulations and a simulation of the dense school on a single Cartesian mesh as CASE III for reference. All three simulations have the same resolution of 0.0022 around the solid boundaries. The TLMR reduces the number of meshes of the 2D examples dramatically from 6.3 million (CASE III) to 1.0 ~ 1.2 million (CASE I and CASE II). The reduction of the number of meshes significantly reduces the time in solving the discretized equations. The iterative solver for the momentum equation on the refinement mesh is 11 ~ 12 times faster than the reference one. The speedup of the iterative Poisson solver can be 10 times, or 6 times if there are blocks intralayer-connected. This is because the iterations often increase when the intralayer connection appears, from an average of 20 steps to 39 in current examples.

Table 4: Runtime information of the 2D fish school swimming, averaged over 100 time steps.

	CASE I	CASE II	CASE III (Global Refinement)
Total gird size ($\times 10^6$)	1.0	1.2	6.3
Iterations of momemtum	10	10	12
Iterations of Poission	20	39	16
Time of synchronization (sec.)	0.01	0.02	-
Time of momentum solver (sec.)	0.89	0.95	10.80[a]
Time of Poisson solver (sec.)	2.84	5.20	29.16[a]
Total time of solving eqns. (sec.)	3.74	6.17	39.96[a]
Mesh speedup, $SoM(n)$	6.3	5.3	1
Speedup, $S(n)$	10.7	6.5	1
Parallel efficiency, $\eta(n)$	0.28	0.20	1

[a]Time of simulating the sparse school problem

To better describe the computation efficiency, we define the speedup of computation as

$$S(n) = \frac{\text{Time of computation on a single Cartesian block with given grid resolution}}{\text{Time of computation on refinement meshes}}, \tag{11}$$

where n is the number of processors, or refinement blocks, used in the mesh refinement. Meanwhile, as the computation is proportional to the total number of meshes, the speedup of computation due to the saving of meshes is defined as

$$SoM(n) = \frac{\text{Number of meshes in a single Cartesian block with given grid resolution}}{\text{Number of meshes in refinement meshes}}. \tag{12}$$

The saving of meshes, or memory, can be computed from the mesh speedup by $1 - 1/SoM(n)$. The parallel computation efficiency, $\eta(n)$, of the present mesh refinement technique computed on a distributed memory system is then assessed by

$$\eta(n) = \frac{S(n)}{SoM(n) \cdot n}. \tag{13}$$

The speedup of current examples with two layers refinement can reach as high as 10.7, in which *SoM* contributes 6.3. The parallel computation efficiency with three blocks, or three processors, reaches 0.28. Meanwhile, the saving in computational memory, or the number of meshes, can reach 84% for the 2D examples. For CASE I and CASE II, the speedup is mainly contributed by the saving of mesh by a percentage of 60% ∼ 80%.

As intralayer communication efficiency can affect the iteration time, we compare the two communication strategies proposed in § 2.3.3. We repeat CASE II using the proposed one-interface and two-interface strategy (the default approach in earlier simulations) for intralayer communication and plot the results in Fig. 15, which also includes the single block case (CASE III) for reference. Fig. 15(a) shows the maximum residual during the iteration. The multigrid algorithm is most efficient on a single Cartesian mesh, CASE III, and the Poisson equation converges in about 20 steps. The two-interface strategy of updating the outer ghost cells and iteratively solving the whole rectangular block slightly increases the iterative steps. However, the one-interface strategy needs much more steps and even thousands to converge, neutralizing the efficiency of the multigrid algorithm. Meanwhile, two strategies yield almost identical pressure as shown in Fig. 15(b). The pressure distributions are also similar to the CASE III on a single block except for the far-field area, which is not sufficiently resolved in the current simulation due to a lack of interest there.

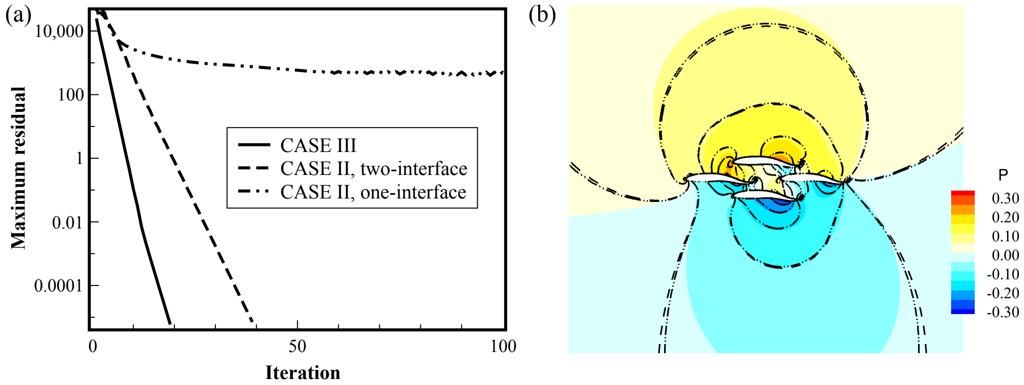

Figure 15: Comparison of the iterative Poisson solver using two communication strategies: (a) the residual during the iterations and (b) the contour plot of the pressure. In figure (b), dash lines are for the CASE II using the two-interface communication strategy, and dash-double-dot lines are for CASE II using the one-interface communication strategy, and color contour plots the results on the global refinement meshes for reference.

3.3.2. *Simulation of the three-dimensional single trout swimming*

The speedup of the TLMR method can be more prominent in a 3D simulation. To demonstrate the efficiency for a 3D flow with moving boundaries, we present the simulation of a rainbow trout (*Oncorhynchus mykiss*). The trout geometry and kinematics are constructed from videos taken from an orthotropic high-speed camera system constituting a lateral, ventral, and posterior view, using the same reconstruction method adopted by Liu et al. [74, 75]. The image snapshots and the reconstruction are illustrated in Fig. 16. The Reynolds number in this numerical investigation is 4800, and the Strouhal number is about 0.46.

The configuration for the simulation is shown in Fig. 17(a), and the local refinement mesh is shown in Fig. 17(b). The domain size is $10L \times 6L \times 6L$, where L is the body length of a trout. A

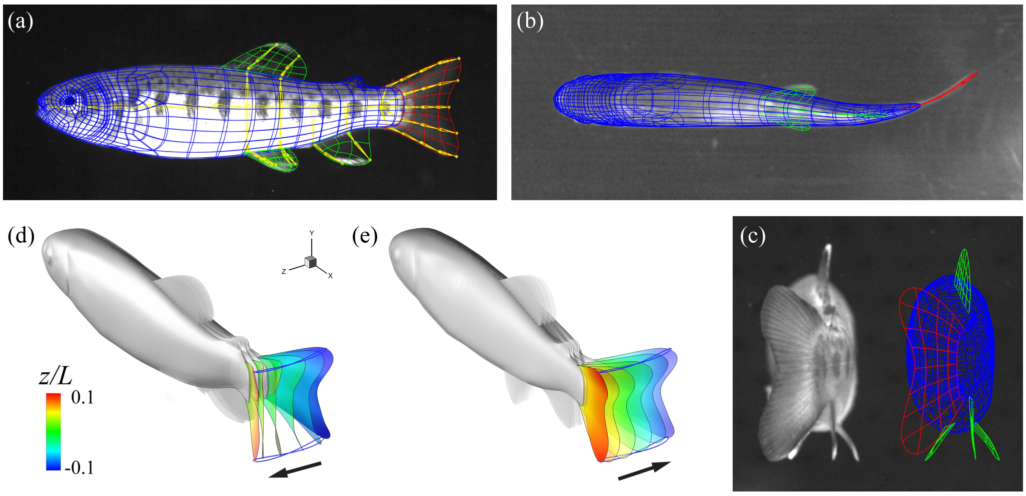

Figure 16: Morphological and kinematical modeling of juvenile rainbow trout (*Oncorhynchus mykiss*) during steady swimming. (a-b) High-speed camera images of live rainbow trout swimming overlapped with computational model from lateral and ventral views, (c) Side-by-side comparison of live trout and computational mode from posterior view (d-e) Reconstructed swimming kinematics of computational model during left-to-right (L-to-R) and right-to-left (R-to-L) strokes, respectively.

block cutting through the body and encompassing all the fins is added as a third layer for better resolution of the flow around the undulating fins. The finest resolution is $1.75 \times 10^{-3}L$ around the posterior part of the trout. The boundary conditions are set the same as in the pitching panel cases.

The wake topology in one swimming cycle is plotted in Fig. 17(c) and Fig. 17(d). Among them, Fig. 17(c1) and (c2) show that strong coherent vortex structures are generated on the leading edge of the fins including the dorsal fin, pelvic fin, anal fin, and caudal fin. These leading-edge vortexes contribute to the majority of thrust production. Besides, in each half period of the tail-beat cycle, a vortex ring is shed from the trailing edge of the caudal fin and connects with neighboring vortex rings. The TLMR method recursively refines meshes around the trout. Therefore, we observe smooth flow velocity from the contour plot in the cut slices around the trout body, as shown in Fig. 17(c2) and (d2). Particularly, we plotted the boundary velocity profile in Fig. 17(c3) and (d3).

Computational efficiency of the trout swimming

To further demonstrate the efficiency of the TLMR method, the runtime information of four cases is listed and compared in Table 5. CASE I computes on a single Cartesian grid without mesh refinement. The grid resolution is 0.007 and the total number of meshes is about 32.8 million. Case II achieves the same resolution using the one-layer refinement, and the corresponding number of mesh is 7.8 million. CASE III further refines the mesh by placing one additional refinement block on top of CASE II, so the grid resolution reaches 0.0035 and the number of mesh is 10.4 million. CASE IV, which is illustrated in Fig. 17(b), further refines CASE III to achieve a resolution of 0.00175 around the undulating tail and caudal fin. The total number of meshes is 21.5 million. The 3D cases are computed using four OpenMP threads for each refinement block, except that the fourth block in CASE IV uses eight threads.

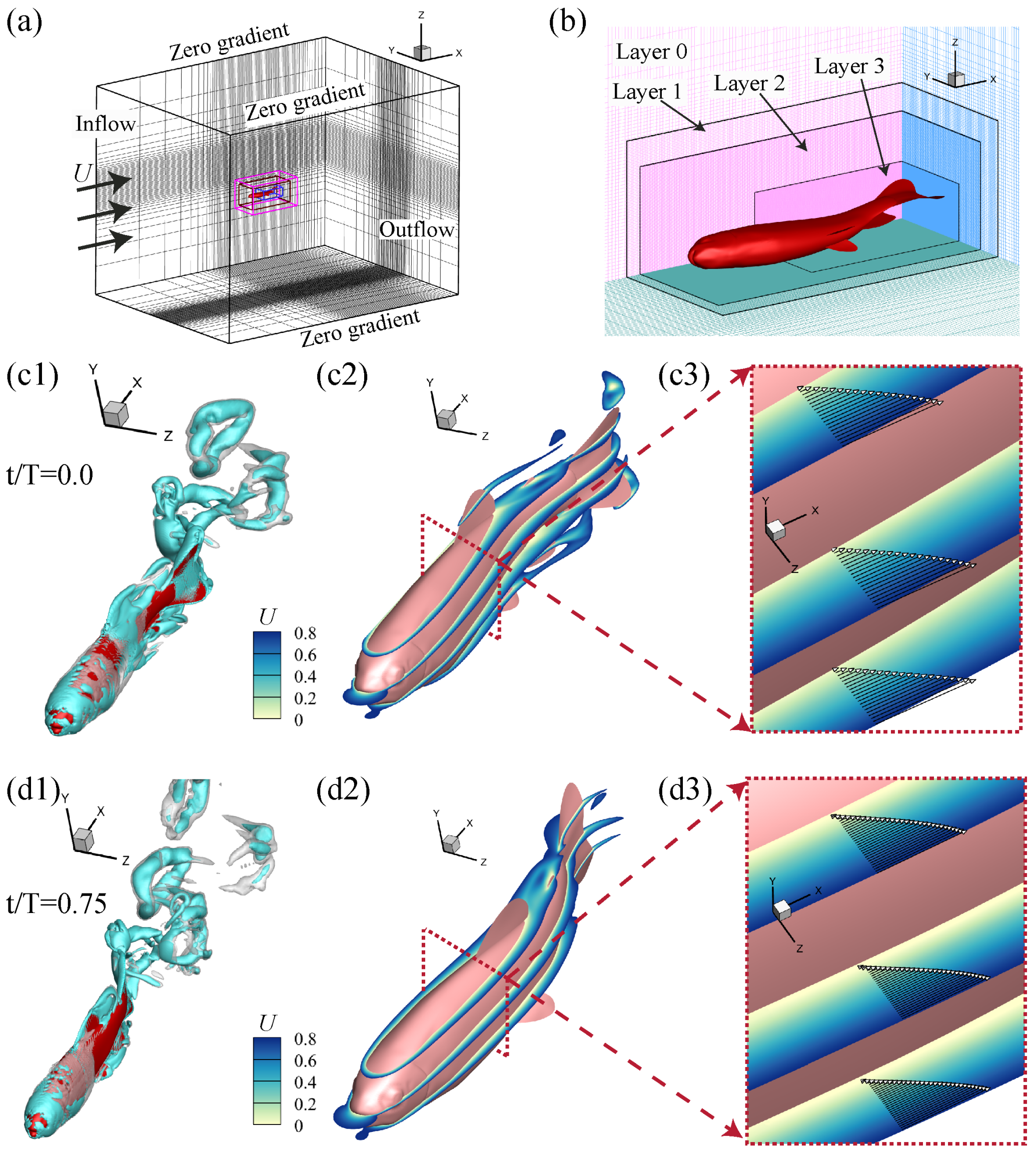

Figure 17: Simulation of trout swimming: (a) setup of the 3D simulation, (b) local mesh refinement around solid boundary, (c1-c3) and (d1-d3) wake after trout, velocity contour on 2D slices, and velocity profile near the boundary when $t = 0T$ or $3/4T$ respectively, where T is the flapping period.

Table 5: Runtime information of the 3D trout simulation, averaged over 100 time steps.

	CASE I Unrefined	CASE II One layer	CASE III Two layers	CASE IV Three layers
Grid resolution ($\Delta = \Delta_x = \Delta_y = \Delta_z$)	0.007	0.007	0.0035	0.00175
Total gird size ($\times 10^6$)	32.8	7.8	11.6	21.5
Total cores used	4	8	12	20
Iterations of momentum	9	8	10	16
Iterations of Poisson	24	15	22	42
Time of synchronization (sec.)	-	0.30	0.63	1.32
Time of momentum solver (sec.)	34.31	4.53	9.91	20.61
Time of Poisson solver (sec.)	379.39	19.15	41.39	127.63
Total time of solving eqns. (sec.)	413.70	23.98	51.93	149.52
Mesh speedup, $SoM(n)$	1	4.2	22.6	97.7
Speedup, $S(n)$	1	17.3	63.8	177.1
Parallel efficiency, $\eta(n)$	1	2.06	0.94	0.45

To compare the computational efficiency, CASE I computed on a single Cartesian grid is chosen as the reference. The speedup $S(n)$ of one layer refinement can reach 17.3, whereas mesh-saving speedup, $SoM(n)$, is around 4.2. The speedup increases significantly when more layers of refinement are applied. For instance, it reaches 63.8 and 177.1 when applied with two and three-layer mesh refinement, respectively, whereas the mesh-saving speedup reaches up to 97.7, accounting for 55% of the total speedup. At the same time, the saving of computational memory for the 3D problem is greater than 76% and reaches a remarkable 99% when three-layer mesh refinement is applied. In the above computation, the reference cases for CASE III and CASE IV are derived from CASE I, whereas the mesh is assumed to be refined from that of CASE I in each direction by a factor of 2 or 4 respectively, and the computation time is assumed to scale linearly with the total number of meshes.

3.3.3. Simulation of the three-dimensional trout school swimming

In this section, we present results from a simulation devised to examine hydrodynamics of fish school swimming with a significant level of complexity. The results presented here are primarily intended to show the complexity of the flow and the ability of the solver to handle a case that includes multiple moving objects with strong interactions. The fish model and flow conditions in § 3.3.2 are adopted here to study body-body and body-wake interactions in a diamond-shaped fish school with a streamwise distance of $0.4L$ and lateral spacing of $0.6L$. The flow simulation is conducted on a $12L \times 6L \times 8L$ computational domain size. A Cartesian grid of size $320 \times 96 \times 224$ and two wrapped TLMR blocks are used around each fish which provides high resolution meshes as shown in Fig. 18(a), where the finest mesh is $3.35 \times 10^{-3}L$. In the specific case presented here, the Reynolds number based on the fish body length is 5430 and the Strouhal number is 0.41. Boundary conditions for the problem are set as shown in Fig. 17(a). The flow simulation uses 24 cores (4 cores per 6 refinement blocks), 96GB of memory, and a total of 576 CPU hours, or 24 hours of wall clock time, for computing one tail-beat cycle. Figure 18(b) demonstrates the flow structures, which are visualized by the isosurfaces of the Q-criterion, at the end of the 4th tail-beat cycle of the fish school, where the isosurface in grey color represents $Q = 1.6$ and the one in blue color is $Q = 4.0$. At this stage, the dominant vortex features in the

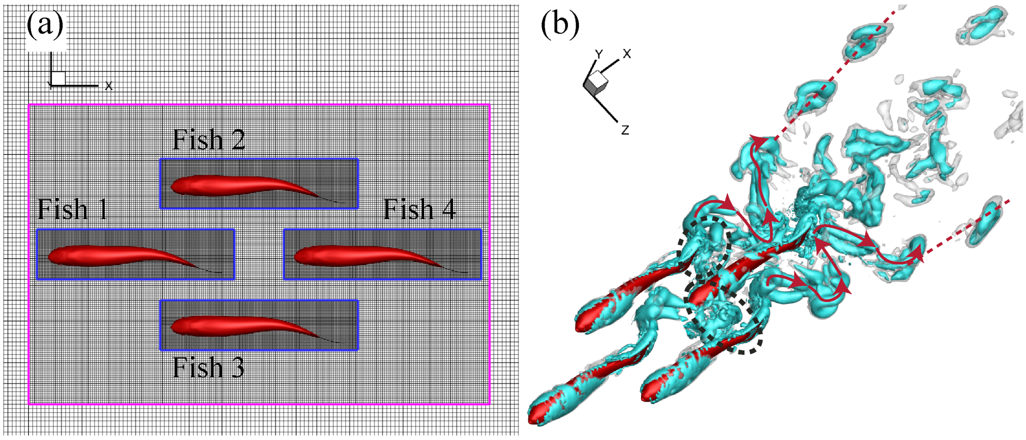

Figure 18: (a) Top view of the computational meshes around four fishes, and (b) Q isosurfaces at the end of the 4th tail-beat of a swimming fish school.

flow are the vortex rings and inter-connected wake vortices created during the stroke. Similar to Fig. 17, each fish sheds two sets of vortex rings downstream. Due to the existence of neighboring fishes, these vortex rings interact with the neighboring fish bodies and merge with other vortices, as shown in the circled regions in Fig. 18(b). Furthermore, due to its special position in the fish school, fish 4 interacts with the vortices shed from fish 1 as well as the half-stroke rings shed from fish 2 and fish 3, respectively. These vortices merge with the vortices produced by the tail of fish 4, then form the strong inter-connected wake vortices. This school formation is being further refined with changing of distance and tail-beat phase difference between all pairs of fish within the school and will then be used for a detailed investigation of fish school hydrodynamics.

4. Conclusions

In this paper, we have developed an efficient IB method using the TLMR for the problems of bio-inspired flows induced by the motion of single or multiple objects that use a Cartesian grid finite difference scheme for the incompressible Navier-Stokes equations. This mesh refinement approach recursively refines regions of interest by subdividing the blocks of Cartesian grids and hence enables hybrid parallelism on a distributed memory system connected by multiple computing nodes with multi-core processors. A key contribution of this paper is that it introduces a new iterative approach to solve the discretized equations on the refinement meshes for faster convergence while preserving numerical accuracy. That is, the momentum equation is discretized on all grids but the refined ones and solved using an iterative algorithm. Meanwhile, the Poisson equation is discretized on all grids and solved recursively from the coarsest block of meshes to the finest ones, of which the boundary values are interpolated and synchronized from the former. Additionally, the discretized equations are solved parallelly using the Schwarz method when the refinement blocks of a coarse one are connected.

Several two- and three-dimensional canonical flows are simulated and the computed results are compared with available data sets to establish the accuracy and fidelity of the new TLMR-IBM flow solver. Numerical examples without or with immersed solid boundaries, such as the Taylor-Green vortex flow and the flow past a circular cylinder, demonstrate that the new solver achieves second-order spatial accuracy for both the velocity and pressure. Simulations are also conducted for flows with a stationary or moving object at different Reynolds numbers ranging from $O(10^2)$ up to $O(10^4)$, and we show that the current solver accurately predicts the drag forces, shedding frequencies, surface pressure and boundary velocity for the canonical flow past stationary cylinder problem and the evolution of the vorticity field that matched well with the 3D PIV data of flows past a pitching plate. The computation of 2D fish school flows shows a good versatility of the TLMR method by wrapping finer mesh layers around multiple objects and a great deal of saving of the computational memory and time up to 80% by the usage of TLMR method. Finally, we demonstrate the ability of the solver to handle extremely complicated three-dimensional moving objects by showing selected results from the body- / fin-fin interactions in trout swimming and a diamond-shaped trout school swimming. These cases show that our solver not only obtains the time-resolved flow field near the fish body and in the far wakes but also permits the use of a much smaller number of meshes that are over 80% less than a global refinement. Consequently, the current solver saves significant computational time for the 3D problem in addition to the parallel computation, and the saving can be larger when more wrapping boxes of refinement meshes instead of a global refinement are used.

Acknowledgements

This work is supported by the ONR MURI Grant Number N00014-14-1-0533, NSF CNS-1931929 and CBET-2027534.

References

[1] C. S. Peskin, The immersed boundary method, Acta Numerica 11 (2002) 479–517.
[2] J. Wang, Y. Ren, C. Li, H. Dong, Computational investigation of wing-body interaction and its lift enhancement effect in hummingbird forward flight, Bioinspiration & Biomimetics 14 (4) (2019) 046010.
[3] Z. Wu, J. Liu, J. Yu, H. Fang, Development of a novel robotic dolphin and its application to water quality monitoring, IEEE/ASME Transactions on Mechatronics 22 (5) (2017) 2130–2140.
[4] J. Wang, D. K. Wainwright, R. E. Lindengren, G. V. Lauder, H. Dong, Tuna locomotion: a computational hydrodynamic analysis of finlet function, Journal of the Royal Society Interface 17 (165) (2020) 20190590.
[5] H. Jasak, Z. Tukovic, Automatic mesh motion for the unstructured finite volume method, Transactions of FAMENA 30 (2) (2006) 1–20.
[6] T. C. Rendall, C. B. Allen, Efficient mesh motion using radial basis functions with data reduction algorithms, Journal of Computational Physics 228 (17) (2009) 6231–6249.
[7] M. Souli, A. Ouahsine, L. Lewin, ALE formulation for fluid–structure interaction problems, Computer methods in applied mechanics and engineering 190 (5-7) (2000) 659–675.
[8] C. S. Peskin, Flow patterns around heart valves: a numerical method, Journal of Computational Physics 10 (2) (1972) 252–271.
[9] M.-C. Lai, C. S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, Journal of Computational Physics 160 (2) (2000) 705–719.
[10] R. Mittal, G. Iaccarino, Immersed boundary methods, Annual Review of Fluid Mechanics 37 (2005) 239–261.
[11] T. Ye, R. Mittal, H. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, Journal of Computational Physics 156 (2) (1999) 209–240.
[12] D. M. Ingram, D. M. Causon, C. G. Mingham, Developments in Cartesian cut cell methods, Mathematics and Computers in Simulation 61 (3-6) (2003) 561–572.
[13] J. Yang, F. Stern, Sharp interface immersed-boundary/level-set method for wave–body interactions, Journal of Computational Physics 228 (17) (2009) 6590–6616.
[14] Z. Alan Wei, Z. Charlie Zheng, X. Yang, Computation of flow through a three-dimensional periodic array of porous structures by a parallel immersed-boundary method, Journal of Fluids Engineering 136 (4) (2014).
[15] N. J. Nair, A. Goza, A strongly coupled immersed boundary method for fluid-structure interaction that mimics the efficiency of stationary body methods, Journal of Computational Physics 454 (2022) 110897.

[16] J. D. Eldredge, A method of immersed layers on cartesian grids, with application to incompressible flows, Journal of Computational Physics 448 (2022) 110716.
[17] J. Mohd-Yusof, Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries, Center for Turbulence Research Annual Research Briefs 161 (1) (1997) 317–327.
[18] Y.-H. Tseng, J. H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, Journal of Computational Physics 192 (2) (2003) 593–623.
[19] R. Mittal, H. Dong, M. Bozkurttas, F. Najjar, A. Vargas, A. Von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, Journal of Computational Physics 227 (10) (2008) 4825–4852.
[20] P. Han, G. V. Lauder, H. Dong, Hydrodynamics of median-fin interactions in fish-like locomotion: Effects of fin shape and movement, Physics of Fluids 32 (1) (2020) 011902.
[21] I. Kossaczkỳ, A recursive approach to local mesh refinement in two and three dimensions, Journal of Computational and Applied Mathematics 55 (3) (1994) 275–288.
[22] M. J. Berger, P. Colella, et al., Local adaptive mesh refinement for shock hydrodynamics, Journal of Computational Physics 82 (1) (1989) 64–84.
[23] J. Zhu, O. Zienkiewicz, Adaptive techniques in the finite element method, Communications in Applied Numerical Methods 4 (2) (1988) 197–204.
[24] P. Solin, K. Segeth, I. Dolezel, Higher-order finite element methods, CRC Press, 2003.
[25] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, Journal of Computational Physics 190 (2) (2003) 572–600.
[26] M. J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, Journal of Computational Physics 53 (3) (1984) 484–512.
[27] J. Bell, M. Berger, J. Saltzman, M. Welcome, Three-dimensional adaptive mesh refinement for hyperbolic conservation laws, SIAM Journal on Scientific Computing 15 (1) (1994) 127–138.
[28] M. J. Berger, R. J. LeVeque, Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems, SIAM Journal on Numerical Analysis 35 (6) (1998) 2298–2316.
[29] E. Steinthorsson, D. Modiano, W. Crutchfield, J. Bell, P. Colella, An adaptive semi-implicit scheme for simulations of unsteady viscous compressible flows, in: 12th Computational Fluid Dynamics Conference, 1995, p. 1727.
[30] D. Graves, P. Colella, D. Modiano, J. Johnson, B. Sjogreen, X. Gao, A Cartesian grid embedded boundary method for the compressible navier–stokes equations, Communications in Applied Mathematics and Computational Science 8 (1) (2013) 99–122.
[31] L. H. Howell, J. B. Bell, An adaptive mesh projection method for viscous incompressible flow, SIAM Journal on Scientific Computing 18 (4) (1997) 996–1013.
[32] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, M. L. Welcome, A conservative adaptive projection method for the variable density incompressible navier–stokes equations, Journal of Computational Physics 142 (1) (1998) 1–46.
[33] M. Vanella, P. Rabenold, E. Balaras, A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid–structure interaction problems, Journal of Computational Physics 229 (18) (2010) 6427–6449.
[34] C. Liu, C. Hu, Block-based adaptive mesh refinement for fluid–structure interactions in incompressible flows, Computer Physics Communications 232 (2018) 104–123.
[35] J. Yang, An easily implemented, block-based fast marching method with superior sequential and parallel performance, SIAM Journal on Scientific Computing 41 (5) (2019) C446–C478.
[36] Y.-F. Peng, R. Mittal, A. Sau, R. R. Hwang, Nested Cartesian grid method in incompressible viscous fluid flow, Journal of Computational Physics 229 (19) (2010) 7072–7101.
[37] X. Deng, H. Dong, A highly efficient sharp-interface immersed boundary method with adaptive mesh refinement for bio-inspired flow simulations, in: APS Division of Fluid Dynamics Meeting Abstracts, 2017, pp. Q30–002.
[38] W. Zhang, Y. Pan, Y. Gong, H. Dong, J. Xi, A versatile IBM-based AMR method for studying human snoring, in: Fluids Engineering Division Summer Meeting, Vol. 85284, American Society of Mechanical Engineers, 2021, p. V001T02A039.
[39] J. Kim, P. Moin, Application of a fractional-step method to incompressible navier-stokes equations, Journal of Computational Physics 59 (2) (1985) 308–323.
[40] D. L. Brown, R. Cortez, M. L. Minion, Accurate projection methods for the incompressible navier–stokes equations, Journal of Computational Physics 2 (168) (2001) 464–499.
[41] H. Dong, R. Mittal, F. Najjar, Wake topology and hydrodynamic performance of low-aspect-ratio flapping foils, Journal of Fluid Mechanics 566 (2006) 309.
[42] H. Dong, M. Bozkurttas, R. Mittal, P. Madden, G. Lauder, Computational modelling and analysis of the hydrodynamics of a highly deformable fish pectoral fin, Journal of Fluid Mechanics 645 (2010) 345.
[43] G. Liu, H. Dong, C. Li, Vortex dynamics and new lift enhancement mechanism of wing–body interaction in insect forward flight, Journal of Fluid Mechanics 795 (2016) 634–651.

[44] Y. Saad, M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (3) (1986) 856–869.
[45] H. A. Van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 13 (2) (1992) 631–644.
[46] H. L. Stone, Iterative solution of implicit approximations of multidimensional partial differential equations, SIAM Journal on Numerical Analysis 5 (3) (1968) 530–558.
[47] G. Schneider, M. Zedan, A modified strongly implicit procedure for the numerical solution of field problems, Numerical Heat Transfer 4 (1) (1981) 1–19.
[48] M. Zedan, G. Schneider, A three-dimensional modified strongly implicit procedure for heat conduction, AIAA Journal 21 (2) (1983) 295–303.
[49] H. A. Schwarz, Ueber einige abbildungsaufgaben., Ges. Math. Abh., Berlin (1869).
[50] P.-L. Lions, On the schwarz alternating method. i, in: First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Vol. 1, Paris, France, 1988, p. 42.
[51] S. R. Fulton, P. E. Ciesielski, W. H. Schubert, Multigrid methods for elliptic problems: A review, Monthly Weather Review 114 (5) (1986) 943–959.
[52] J. H. Bramble, Multigrid methods, Chapman and Hall/CRC, 2019.
[53] R. Wienands, W. Joppich, Practical Fourier analysis for multigrid methods, Chapman and Hall/CRC, 2004.
[54] J. Green, P. Jimack, A. Mullis, J. Rosam, Towards a three-dimensional parallel, adaptive, multilevel solver for the solution of nonlinear, time-dependent, phase-change problems, Parallel, Distributed and Grid Computing for Engineering 21 (2009) 251–274.
[55] P. MacNeice, K. M. Olson, C. Mobarry, R. De Fainchtein, C. Packer, Paramesh: A parallel adaptive mesh refinement community toolkit, Computer Physics Communications 126 (3) (2000) 330–354.
[56] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, et al., AMReX: a framework for block-structured adaptive mesh refinement, Journal of Open Source Software 4 (37) (2019) 1370–1370. doi:10.21105/joss.01370.
[57] T. Guillet, R. Teyssier, A simple multigrid scheme for solving the poisson equation with arbitrary domain boundaries, Journal of Computational Physics 230 (12) (2011) 4756–4771.
[58] G. I. Taylor, A. E. Green, Mechanism of the production of small eddies from large ones, Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences 158 (895) (1937) 499–521.
[59] C. P. Ellington, The aerodynamics of hovering insect flight. I. The quasi-steady analysis, Philosophical Transactions of the Royal Society of London. B, Biological Sciences 305 (1122) (1984) 1–15.
[60] R. Tedrake, Z. Jackowski, R. Cory, J. W. Roberts, W. Hoburg, Learning to fly like a bird, in: 14th International Symposium on Robotics Research. Lucerne, Switzerland, Citeseer, 2009.
[61] G. S. Triantafyllou, M. Triantafyllou, M. Grosenbaugh, Optimal thrust development in oscillating foils with application to fish propulsion, Journal of Fluids and Structures 7 (2) (1993) 205–224.
[62] S. Singh, S. Mittal, Flow past a cylinder: shear layer instability and drag crisis, International Journal for Numerical Methods in Fluids 47 (1) (2005) 75–98.
[63] D. J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds numbers, Journal of Fluid Mechanics 6 (4) (1959) 547–567.
[64] R. D. Henderson, Details of the drag curve near the onset of vortex shedding, Physics of Fluids 7 (9) (1995) 2102–2104.
[65] P. B. Beaudan, Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number, Ph.D. thesis, Stanford University (1995).
[66] A. Roshko, Experiments on the flow past a circular cylinder at very high Reynolds number, Journal of Fluid Mechanics 10 (3) (1961) 345–356.
[67] C. H. K. Williamson, Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers, Journal of Fluid Mechanics 206 (1989) 579–627. doi:10.1017/S0022112089002429.
[68] C. H. K. Williamson, Three-dimensional wake transition, Journal of Fluid Mechanics 328 (1996) 345–407. doi:10.1017/S0022112096008750.
[69] H.-w. Hsu, Numerical solution of laminar compressible flow over a circular cylinder, Master's thesis, Georgia Institute of Technology (1972).
[70] H. Schlichting, K. Gersten, Boundary layer theory, Springer, 2015.
[71] J. T. King, R. Kumar, M. A. Green, Experimental observations of the three-dimensional wake structures and dynamics generated by a rigid, bioinspired pitching panel, Physical Review Fluids 3 (3) (2018) 034701.
[72] J. C. Hunt, A. A. Wray, P. Moin, Eddies, streams, and convergence zones in turbulent flows, Studying Turbulence Using Numerical Simulation Databases, 2. Proceedings of the 1988 Summer Program (1988).
[73] Y. Pan, H. Dong, Computational analysis of hydrodynamic interactions in a high-density fish school, Physics of Fluids 32 (12) (2020) 121901.
[74] G. Liu, Y. Ren, H. Dong, O. Akanyeti, J. C. Liao, G. V. Lauder, Computational analysis of vortex dynamics

and performance enhancement due to body–fin and fin–fin interactions in fish-like locomotion, Journal of Fluid Mechanics 829 (2017) 65–88.
[75] G. Liu, B. Geng, X. Zheng, Q. Xue, H. Dong, G. V. Lauder, An image-guided computational approach to inversely determine in vivo material properties and model flow-structure interactions of fish fins, Journal of Computational Physics 392 (2019) 578–593.