A Conservative High-Order Method Utilizing Dynamic Transfinite Mortar Elements for Flow Simulations on Curved Nonconforming Sliding Meshes

Bin Zhang*, Chunlei Liang

Department of Mechanical and Aerospace Engineering, The George Washington University, Washington, DC 20052 USA

Abstract

We introduce two concepts in this work: polynomial mortar and transfinite mortar, and apply them to curved nonconforming sliding meshes. It is shown that, on curved meshes, polynomial mortar always carries geometric errors while transfinite mortar has no such errors, which makes the latter superior to the former in almost every aspect. For example, the latter has better accuracies and introduces smaller numerical disturbances. The proposed sliding-mesh method utilizing transfinite mortar is conservative, arbitrarily high-order accurate in space, retains the order of accuracy of a time marching scheme, does not change flow characteristics, and has minimum sliding-speed effects. It is by far the most accurate and most thoroughly studied sliding-mesh method. This method can be extended and applied to a wide range of flow problems, such as propellers, wind turbines, stirred tanks, etc.

Keywords: high-order method, nonconforming mesh, sliding mesh, polynomial mortar, transfinite mortar

1. Introduction

Sliding-mesh methods have been widely used in flow simulations about moving objects. For example, they are ideal choices for handling rotational geometries such as stirred tanks [1] and helicopter rotor blades [2]. They can also be used to ensure good mesh qualities in circumstances where purely deforming mesh may otherwise be very skewed, such as simulations of oscillating wings [3] and vortex-induced-vibration devices [4]. In many applications, a sliding mesh method has advantages over other methods such as overset mesh methods [5] and immersed boundary methods [6] for its simplicity, efficiency and accuracy. However, so far, sliding mesh methods are still mostly limited to low-order (second order and below) schemes that are unfavorable for simulating vortex dominated flows due to large numerical dissipations and dispersions.

Tremendous progress has been made on high-order methods in the past decades in the computational fluid dynamics community [7]. For instance, such methods include the discontinuous Galerkin (DG) method [8, 9], the spectral element method [10–12], the spectral volume method [13, 14], the spectral difference (SD) method [15–19], to name just a few. Among these methods, the SD method solves equations in differential form directly, and is one of the most efficient high-order methods. Recently, the ideas of collocating solution and flux points of the SD method and correcting fluxes using higher-degree polynomials have led to an even more efficient high-order method — the flux reconstruction (FR) method [20, 21], also known as the correction procedure via reconstruction (CPR) method [22]. Besides its better efficiency, by choosing different correction polynomials, the FR method can recover many existing high-order schemes such as DG and SD, and can even produce new schemes that were never reported before. The stability of the FR method has been proved in [23]. The most recent developments on the FR method are summarized in [24].

With more and more applications of high-order methods to flow simulations than ever before, there is a natural need to extend the sliding mesh concept to high-order methods to tackle complex flow problems such as those mentioned above. Ferrer and Willden [25] developed a high-order sliding-mesh DG method based on modal basis functions for simulating incompressible flow problems. Ramírez et al. [26] applied

*Corresponding author. The authors' present address: Clarkson University, Potsdam, NY 13699
Email address: bzh@gwmail.gwu.edu (Bin Zhang)

moving-least-squares stencils to the development of a high-order sliding-mesh finite volume method. The authors of the present work extended the straight stationary mortar concept [27–29] to a curved dynamic mortar concept and developed a simple and efficient high-order sliding-mesh SD method [30], and also extended this method to sliding-deforming meshes [31] and to handle 3D geometries [32]. These methods, however, require uniform mesh on a sliding interface, which restricts mesh generation. Our recent efforts have completely lifted this restriction, and the resulting general nonuniform sliding-mesh method has been demonstrated on the FR method [33] and the SD method for hybrid grids [34]. These methods have also been successfully applied to several flow problems. For example, flows over rotating cylinders of different cross-sectional shapes [35], flapping wing for energy harvesting [36], and more recently the first high-order eddy-resolving simulation of flow over a marine propeller[37]. The dynamic mortar concept has also been applied to automatic mesh refinement for shock capturing on dynamic meshes [38].

Our initial effort in [33] showed that the reduction of geometric errors can potentially make a sliding-mesh method arbitrarily high-order accurate. In this work, we further the investigation to give two new concepts and to provide a more in-depth study on: the spatial and temporal accuracies, the conservation property, the outflow property, the free-stream preservation property, the sliding speed effects, the singularity issue, and the capability of handling multiple objects at the same time. Meanwhile, detailed implementation steps and extension to 3D are also discussed in this work.

The rest of this paper is organized as follows. In Section 2, we briefly describe the equations that are going to be solved numerically. Sections 3 and 4 present the numerical methods, including the FR method, the new concepts, and the sliding-mesh method. Verifications and applications are reported in Section 5. Finally, Section 6 concludes this paper.

2. The Flow Equations

We numerically solve the two-dimensional Navier-Stokes equations in the following conservative form,

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{0}, \tag{1}$$

where $\mathbf{Q}$ is the vector of conservative variables; $\mathbf{F}$ and $\mathbf{G}$ are the flux vectors in the x- and the y-direction, respectively. Their expressions are

$$\mathbf{Q} = [\rho \ \rho u \ \rho v \ E]^{\mathsf{T}}, \tag{2}$$

$$\mathbf{F} = \mathbf{F}_{\text{inv}}(\mathbf{Q}) + \mathbf{F}_{\text{vis}}(\mathbf{Q}, \nabla\mathbf{Q}), \tag{3}$$

$$\mathbf{G} = \mathbf{G}_{\text{inv}}(\mathbf{Q}) + \mathbf{G}_{\text{vis}}(\mathbf{Q}, \nabla\mathbf{Q}), \tag{4}$$

where ρ is fluid density, u and v are the Cartesian velocity components, and E is the total energy per volume defined as

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2), \tag{5}$$

where p is pressure, and γ is the ratio of specific heats and is set to 1.4 in this work for ideal gas. The inviscid and the viscous flux vectors are

$$\mathbf{F}_{\text{inv}} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E+p)u \end{bmatrix}, \quad \mathbf{G}_{\text{inv}} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E+p)v \end{bmatrix}, \tag{6}$$

$$\mathbf{F}_{\text{vis}} = -\begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ u\tau_{xx} + v\tau_{yx} + \kappa T_x \end{bmatrix}, \quad \mathbf{G}_{\text{vis}} = -\begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} + \kappa T_y \end{bmatrix}, \tag{7}$$

where

$$\tau_{ij} = \mu(u_{i,j} + u_{j,i}) + \lambda\delta_{ij}u_{k,k},$$

is the shear stress tensor, μ is the dynamic viscosity, $\lambda = -2/3\mu$ based on the Stokes' hypothesis, δ_{ij} is the Kronecker delta, κ is the thermal conductivity, and T is temperature which is related to density and pressure through the ideal gas law

$$p = \rho \mathcal{R} T, \tag{8}$$

where $\mathcal{R}$ is the gas constant.

To deal with moving grid, we employ an arbitrary-Lagrangian-Eulerian (ALE) approach, and map a moving physical domain to a fixed computational domain. Let (t, x, y) denote the physical time and space, and (τ, ξ, η) the computational ones. Further assume that the mapping is: $t = \tau$, $x = x(\tau, \xi, \eta)$, and $y = y(\tau, \xi, \eta)$. It can be shown that the flow equations will take the following conservative form in the computational space

$$\frac{\partial \widetilde{\mathbf{Q}}}{\partial t} + \frac{\partial \widetilde{\mathbf{F}}}{\partial \xi} + \frac{\partial \widetilde{\mathbf{G}}}{\partial \eta} = \mathbf{0}, \tag{9}$$

where $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{F}}$, and $\widetilde{\mathbf{G}}$ are the computational solution vector and flux vectors, and they are related to the physical ones as

$$\widetilde{\mathbf{Q}} = |\mathcal{J}|\mathbf{Q}, \tag{10}$$

$$\widetilde{\mathbf{F}} = (-x_t y_\eta + y_t x_\eta)\mathbf{Q} + y_\eta \mathbf{F} - x_\eta \mathbf{G}, \tag{11}$$

$$\widetilde{\mathbf{G}} = (\ \ x_t y_\xi - y_t x_\xi)\mathbf{Q} - y_\xi \mathbf{F} + x_\xi \mathbf{G}. \tag{12}$$

Alternatively, let Q, F, $\widetilde{F}$, etc., each denote a component (at the same position) of the corresponding boldface vector, and then the relations in (10)-(12) can be written in the following matrix form

$$\begin{bmatrix} \widetilde{Q} \\ \widetilde{F} \\ \widetilde{G} \end{bmatrix} = |\mathcal{J}|\mathcal{J}^{-1} \begin{bmatrix} Q \\ F \\ G \end{bmatrix}, \tag{13}$$

where $\mathcal{J}$ represents the Jacobian matrix, $|\mathcal{J}|$ is its determinant, and $\mathcal{J}^{-1}$ is the inverse Jacobian matrix. These metric terms have the following expressions,

$$\mathcal{J} = \frac{\partial(t, x, y)}{\partial(\tau, \xi, \eta)} = \begin{bmatrix} 1 & 0 & 0 \\ x_\tau & x_\xi & x_\eta \\ y_\tau & y_\xi & y_\eta \end{bmatrix}, \qquad |\mathcal{J}| = x_\xi y_\eta - x_\eta y_\xi, \tag{14}$$

$$\mathcal{J}^{-1} = \frac{\partial(\tau, \xi, \eta)}{\partial(t, x, y)} = \begin{bmatrix} 1 & 0 & 0 \\ \xi_t & \xi_x & \xi_y \\ \eta_t & \eta_x & \eta_y \end{bmatrix} = \frac{1}{|\mathcal{J}|} \begin{bmatrix} |\mathcal{J}| & 0 & 0 \\ -x_t y_\eta + y_t x_\eta & y_\eta & -x_\eta \\ x_t y_\xi - y_t x_\xi & -y_\xi & x_\xi \end{bmatrix}. \tag{15}$$

3. Flux Reconstruction Method on Moving Grid

3.1. Grid Mapping

The first step of the FR method is to map each physical grid element to a standard computational element. In the present implementation, we discretize a computational domain into non-overlapping quadrilateral elements, and employ the following iso-parametric mapping [39] to map each grid element to a unit square element (i.e., $0 \le \xi,\ \eta \le 1$) in the computational space,

$$\mathbf{x}(t, \xi, \eta) = \begin{bmatrix} x(t, \xi, \eta) \\ y(t, \xi, \eta) \end{bmatrix} = \sum_{i=1}^{K} M_i(\xi, \eta) \begin{bmatrix} x_i(t) \\ y_i(t) \end{bmatrix}, \tag{16}$$

where K is the total number of nodes used to approximate a physical element, M_i (see [39] for detailed expressions) and (x_i, y_i) are the shape function and the coordinates of the i-th node, respectively. Figure 1 is a schematic of the iso-parametric representations of a curved physical element using different number of

nodes. Higher-order elements (larger K's) obviously represent curved boundaries more accurately. For this reason, high-order elements should be used along curved boundaries for better accuracy.

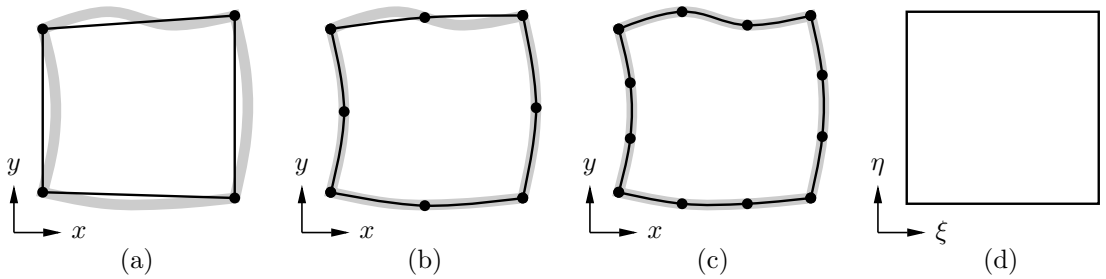


Figure 1: Iso-parametric representation (black lines) of a curved element (gray lines): (a) $K = 4$ (linear), (b) $K = 8$ (quadratic), (c) $K = 12$ (cubic), (d) a computational element (unit square).

3.2. Construction of Solution and Flux Polynomials

Within each computational element, solution points (SPs) and flux points (FPs) are defined. The SPs are distributed along each coordinate direction inside the element, and the FPs are distributed along the boundaries only. Figure 2 is a schematic of the distribution of these points for a third-order FR scheme. In this work, the N SPs/FPs in each direction of an N-th order scheme are chosen as the roots of the N-th Legendre polynomial (namely, N Legendre points). At the SPs, Lagrange interpolation bases are defined. For example, the basis at the i-th SP along the ξ direction is

$$h_i(\xi) = \prod_{s=1, s\neq i}^{N} \left(\frac{\xi - X_s}{X_i - X_s} \right), \tag{17}$$

where X_i and X_s are the ξ-coordinates of the i-th and the s-th SP, respectively. If we denote the space of all polynomials of degrees less than or equal to N as $\mathbf{P}_N$, then $h_i \in \mathbf{P}_{N-1}$. Moreover, the h_i's are linearly independent and form a basis for $\mathbf{P}_{N-1}$.

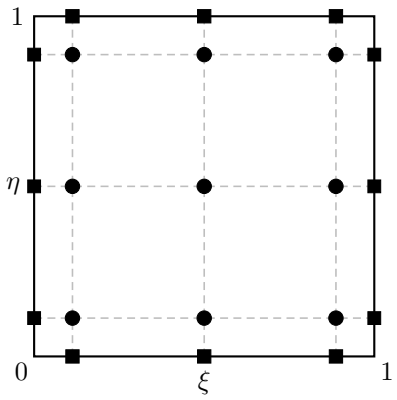


Figure 2: Schematic of the distribution of SPs (circular dots) and FPs (square dots) for a third-order ($P = 2$) FR scheme.

For a function ϕ defined within a mesh element, assume it is smooth (e.g., $\phi \in \mathbf{C}_\infty$) and has discrete values: ϕ_{ij} at (x_i, y_j) that corresponds to (X_i, X_j) in the computational space, where $i, j = 1, 2, \cdots, N$. This function can then be approximated by the following tensor-product polynomial in $\mathbf{P}_{N-1,N-1} = \mathbf{P}_{N-1} \otimes \mathbf{P}_{N-1}$,

$$\phi(\xi, \eta) = \sum_{j=1}^{N} \sum_{i=1}^{N} \phi_{ij} h_i(\xi) h_j(\eta), \tag{18}$$

where the truncation error for the approximation is $\mathcal{O}(\xi^N, \eta^N)$ based on Taylor series expansion.

Applying (18) to the solution and fluxes, we can obtain the following polynomial representations,

$$\widetilde{\mathbf{Q}}(\xi,\eta) = \sum_{j=1}^{N}\sum_{i=1}^{N} \widetilde{\mathbf{Q}}_{ij} h_i(\xi) h_j(\eta), \tag{19}$$

$$\widetilde{\mathbf{F}}(\xi,\eta) = \sum_{j=1}^{N}\sum_{i=1}^{N} \widetilde{\mathbf{F}}_{ij} h_i(\xi) h_j(\eta), \tag{20}$$

$$\widetilde{\mathbf{G}}(\xi,\eta) = \sum_{j=1}^{N}\sum_{i=1}^{N} \widetilde{\mathbf{G}}_{ij} h_i(\xi) h_j(\eta), \tag{21}$$

where the $()_{ij}$'s are the discrete values at the (i,j)-th SP in a standard element.

The polynomials in (19)-(21) are continuous within each element, but discontinuous across element interfaces (or boundaries). For this reason, common values need to be defined at the interfaces. For instance, the common solution is computed as

$$\mathbf{Q}^{\text{com}} = \frac{1}{2}(\mathbf{Q}^{\text{L}} + \mathbf{Q}^{\text{R}}), \tag{22}$$

where $\mathbf{Q}^{\text{L}}$ and $\mathbf{Q}^{\text{R}}$ are the solution vectors on the left side and the right side of an interface, respectively.

To compute the common (normal) inviscid flux, we employ a Riemann solver, such as the following Rusanov solver [40] with modification for moving mesh,

$$\mathbf{F}^{\text{com}}_{\text{inv}} = \frac{1}{2}\big[(\overleftrightarrow{\mathbf{F}}^{\text{L}}_{\text{inv}} + \overleftrightarrow{\mathbf{F}}^{\text{R}}_{\text{inv}}) \cdot \mathbf{n} - \lambda(\mathbf{Q}^{\text{R}} - \mathbf{Q}^{\text{L}})\big] - (\mathbf{v}_g \cdot \mathbf{n})\mathbf{Q}^{\text{com}}, \tag{23}$$

where $\overleftrightarrow{\mathbf{F}}^{\text{L}}_{\text{inv}} = (\mathbf{F}^{\text{L}}_{\text{inv}}, \mathbf{G}^{\text{L}}_{\text{inv}})$ and $\overleftrightarrow{\mathbf{F}}^{\text{R}}_{\text{inv}} = (\mathbf{F}^{\text{R}}_{\text{inv}}, \mathbf{G}^{\text{R}}_{\text{inv}})$ are the inviscid flux vectors on the two sides of an interface; $\mathbf{n} = \mathbf{N}/\|\mathbf{N}\|$ is the unit normal vector with

$$\mathbf{N} = (y_\eta, -x_\eta) \text{ or } (-y_\xi, x_\xi) \tag{24}$$

depending on which direction (i.e., ξ or η) the interface is mapped to; λ is the largest characteristic speed with the following expression,

$$\lambda = |V_n| + c = \Big|\big(\frac{1}{2}(\mathbf{v}^{\text{L}} + \mathbf{v}^{\text{R}}) - \mathbf{v}_g\big) \cdot \mathbf{n}\Big| + c, \tag{25}$$

where $\mathbf{v}^{\text{L}} = (u^{\text{L}}, v^{\text{L}})$ and $\mathbf{v}^{\text{R}} = (u^{\text{R}}, v^{\text{R}})$ are flow velocities on the two sides of an interface; $\mathbf{v}_g = (x_t, y_t)$ represents grid velocity; c is the local speed of sound. The physical normal flux in (23) is converted to a computational one by multiplying the magnitude of normal, i.e.,

$$\widetilde{\mathbf{F}}^{\text{com}}_{\text{inv}} \text{ or } \widetilde{\mathbf{G}}^{\text{com}}_{\text{inv}} = \|\mathbf{N}\|\mathbf{F}^{\text{com}}_{\text{inv}}. \tag{26}$$

The common viscous flux is calculated from the common solution and common gradient,

$$\mathbf{F}^{\text{com}}_{\text{vis}} = \mathbf{F}_{\text{vis}}(\mathbf{Q}^{\text{com}}, (\nabla\mathbf{Q})^{\text{com}}), \tag{27}$$

where the common gradient is the average of the left and the right values, i.e.,

$$(\nabla\mathbf{Q})^{\text{com}} = ((\nabla\mathbf{Q})^{\text{L}} + (\nabla\mathbf{Q})^{\text{R}})/2. \tag{28}$$

The procedure for calculating $\nabla\mathbf{Q}$ will be briefly discussed in the next section. The common viscous flux in (27) is converted to a computational one in the same way as (26).

3.3. Flux Reconstruction

The spatial derivatives in (9) reduce the two flux terms to $\mathbf{P}_{N-2,N-1}$ and $\mathbf{P}_{N-1,N-2}$, respectively, making them inconsistent with the solution term which is in $\mathbf{P}_{N-1,N-1}$. To overcome this issue, the flux polynomials need to be reconstructed to be at least $\mathbf{P}_{N,N-1}$ and $\mathbf{P}_{N-1,N}$.

To do this, we use higher degree correction functions/polynomials. The corrected/reconstructed fluxes take the following forms,

$$\widehat{\mathbf{F}}(\xi,\eta) = \widetilde{\mathbf{F}}(\xi,\eta) + [\widetilde{\mathbf{F}}^{\text{com}}(0,\eta) - \widetilde{\mathbf{F}}(0,\eta)] \cdot g_{\text{L}}(\xi) + [\widetilde{\mathbf{F}}^{\text{com}}(1,\eta) - \widetilde{\mathbf{F}}(1,\eta)] \cdot g_{\text{R}}(\xi), \tag{29}$$

$$\widehat{\mathbf{G}}(\xi,\eta) = \widetilde{\mathbf{G}}(\xi,\eta) + [\widetilde{\mathbf{G}}^{\text{com}}(\xi,0) - \widetilde{\mathbf{G}}(\xi,0)] \cdot g_{\text{L}}(\eta) + [\widetilde{\mathbf{G}}^{\text{com}}(\xi,1) - \widetilde{\mathbf{G}}(\xi,1)] \cdot g_{\text{R}}(\eta), \tag{30}$$

where $\widetilde{\mathbf{F}}(\xi,\eta)$ and $\widetilde{\mathbf{G}}(\xi,\eta)$ are the original flux polynomials from (20) and (21); g_{L} and g_{R} are the left and the right correction functions with degrees no less than N, and they are required to at least satisfy

$$\begin{aligned} g_{\text{L}}(0) = 1, \quad g_{\text{L}}(1) = 0, \\ g_{\text{R}}(0) = 0, \quad g_{\text{R}}(1) = 1, \end{aligned} \tag{31}$$

which ensures that

$$\widehat{\mathbf{F}}(0,\eta) = \widetilde{\mathbf{F}}^{\text{com}}(0,\eta), \quad \widehat{\mathbf{F}}(1,\eta) = \widetilde{\mathbf{F}}^{\text{com}}(1,\eta), \tag{32}$$

$$\widehat{\mathbf{G}}(\xi,0) = \widetilde{\mathbf{G}}^{\text{com}}(\xi,0), \quad \widehat{\mathbf{G}}(\xi,1) = \widetilde{\mathbf{G}}^{\text{com}}(\xi,1), \tag{33}$$

i.e., the reconstructed fluxes still take the common values at cell interfaces. Huynh [20] proposed several correction functions, and we employ the g_{DG} correction function in this study.

Note that to calculate solution gradients consistently, the correction procedure is also applied to the solution polynomial along each direction (only for calculating the gradients). In this way, the resulting gradient polynomials are in $\mathbf{P}_{N-1,N-1}$ as well.

3.4. Time Marching

With proper boundary conditions applied, the discretized equations can be written in the following residual form,

$$\left.\frac{\partial \widetilde{\mathbf{Q}}}{\partial t}\right|_{ij} = -\left[\frac{\partial \widehat{\mathbf{F}}}{\partial \xi} + \frac{\partial \widehat{\mathbf{G}}}{\partial \eta}\right]_{ij} = \mathfrak{R}_{ij}, \quad i,j = 1,2,\cdots,N, \tag{34}$$

where $\mathfrak{R}_{ij}$ is the residual at the (i,j)-th SP. This system can be time marched using either explicit or implicit schemes. In this work, we employ several of the explicit strong stability preserving (SSP) Runge-Kutta schemes reported in [41–43] for the purpose. Furthermore, in this work, all boundary conditions are weakly imposed to increase stability, and more details can be found in, e.g., [15].

3.5. Free-Stream Preservation

Ideally, a moving grid should not disturb a flow field. The simplest situation is that a free-stream flow must stay constant all the time on a moving grid. This is called free-stream preservation. By substituting a constant flow solution into the flow equations in (9), we can get a system of equations that are purely about the geometrics, and are thus known as the geometric conservation law (GCL) [44],

$$\begin{cases} \dfrac{\partial(|\mathcal{J}|\xi_x)}{\partial \xi} + \dfrac{\partial(|\mathcal{J}|\eta_x)}{\partial \eta} = 0, & (35) \\ \dfrac{\partial(|\mathcal{J}|\xi_y)}{\partial \xi} + \dfrac{\partial(|\mathcal{J}|\eta_y)}{\partial \eta} = 0, & (36) \\ \dfrac{\partial|\mathcal{J}|}{\partial t} + \dfrac{\partial(|\mathcal{J}|\xi_t)}{\partial \xi} + \dfrac{\partial(|\mathcal{J}|\eta_t)}{\partial \eta} = 0. & (37) \end{cases}$$

To numerically satisfy free-stream preservation, the same numerical schemes for discretizing the flow equations must be applied to the GCL equations. Since the spatial discretization operator in the FR method

is direct differentiation (which is exact), and the geometric terms are from analytical mapping, the first two GCL equations are hence automatically satisfied. But the third GCL equation generally can not be satisfied automatically. This problem comes from the temporal discretization, for instance, a multiple-stage Runge-Kutta scheme, which is not exact. To overcome this issue, we treat $|\mathcal{J}|$ as an unknown, and solve the third equation numerically for it. The numerical $|\mathcal{J}|$ is then used to update the physical solution according to (10). In this way, the GCL is numerically satisfied, and free-stream preservation is ensured. Similar approach was reported in, e.g., [45].

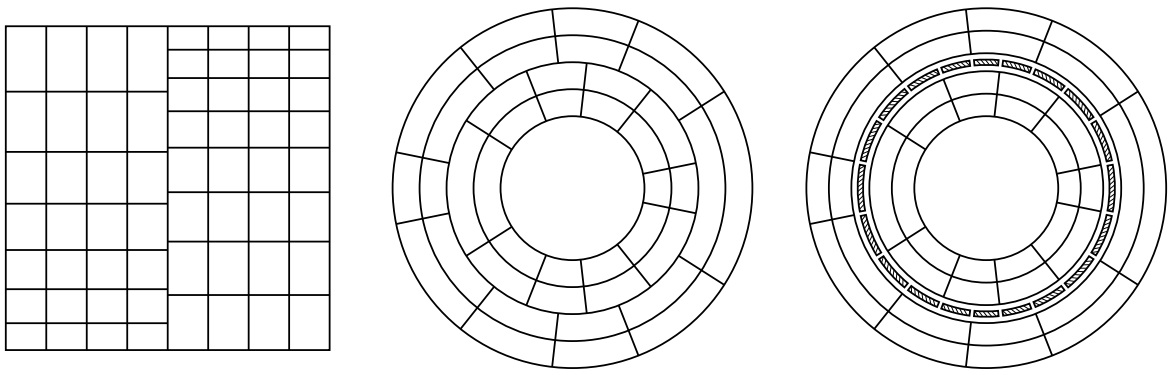
4. A Nonconforming Sliding-Mesh Method

In this section, we first introduce the concepts of sliding mesh and mortar element. Following that, the definitions of polynomial mortar and transfinite mortar are given, with a more detailed discussion on the latter. The projection procedures between cell faces and mortars are described subsequently. Implementation procedures are also provided for completeness. Finally, the extension to 3D are discussed.

4.1. Sliding Mesh and Mortar Elements

The complexity of a sliding mesh depends on the shape of the corresponding sliding interface. The two most fundamental yet most widely used sliding interfaces are the straight one and the circular one as illustrated in Fig. 3(a) and (b), respectively. For simplicity, each sliding mesh in the figure only involves two non-overlapping subdomains. The subdomains are allowed to have relative motions, resulting in dynamically nonconforming meshes. To ensure continuity of solution and conservation of fluxes across a sliding interface, we employ mortar [27] as the communicator between two neighboring subdomains.

Since the straight type has already been well studied, we therefore mainly focus on the circular one in this work. The distribution of mortar elements along the circular sliding interface is sketched in Fig. 3(c). A mortar element is formed between two successive points on the sliding interface. At every time instant, a mortar element is connected to a cell face on its left and a cell face on its right (from a counterclockwise perspective). A cell face is connected to one or multiple mortars on its one side. We call this information as the cell face and mortar connectivities. Note that these connectivities are time-dependent, and need to be updated at every stage of a time marching scheme.



(a) (b) (c)

Figure 3: Sliding mesh with (a) straight interface, (b) circular interface; (c) distribution of mortar elements (hatched lines) along a circular sliding interface.

4.2. Two Mortar Types

Each sliding cell face is mapped to a unit straight line element (e.g., $0 \leq \xi \leq 1$) when the underlying cell is mapped to a unit square element in the computational space. Similarly, we also map each mortar element to a unit straight line element $0 \leq z \leq 1$ in the mortar space to facilitate the construction of, for example, solution and flux polynomials on the mortar. This process has been illustrated in Fig. 4, where Ω denotes a cell face, and the Ξ's denote the associated mortar elements.

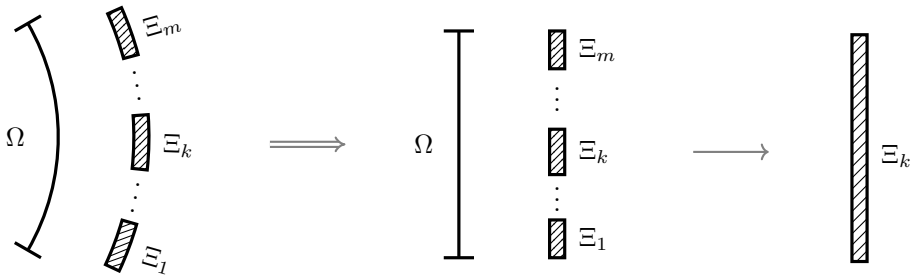


Figure 4: Mapping of a cell face and its mortars to unit line elements: left, physical space; middle, computational space; right, mortar space.

There are two ways to represent a mortar element (and the associated cell face): polynomial approximation (such as iso-parametric mapping) and exact expression (transfinite mapping [46, 47]). We define the resulting mortar elements as polynomial mortar and transfinite mortar, respectively. For traditional straight interface (see Fig. 3(a)), these two representations are equivalent. Therefore, a linear (straight) polynomial mortar element is also the simplest transfinite mortar element. However, for a curved mortar, such as the circular ones in Fig. 3(c), the geometric errors from a polynomial approximation will always pose challenges on conservation, accuracy, etc. Transfinite mortar elements, which carry no geometric errors, can completely avoid these issues.

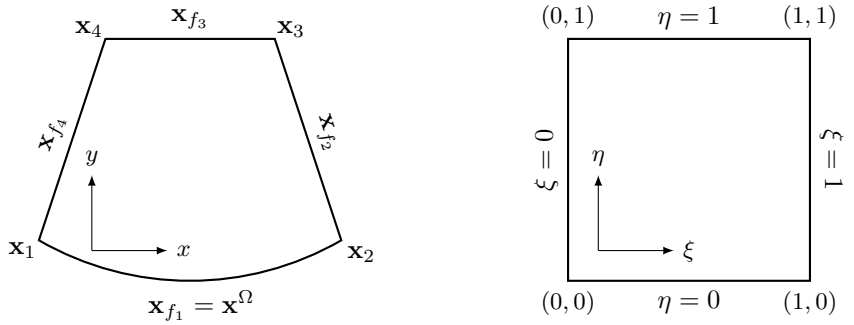


Figure 5: Transfinite mapping of a physical element to a unit square element.

Figure 5 is a schematic of a physical element and the corresponding computational element. The transfinite mapping between these two elements can be expressed as

$$\begin{aligned}\mathbf{x}(t,\xi,\eta) = {} & (1-\eta)\mathbf{x}_{f_1}(t,\xi) + \xi\mathbf{x}_{f_2}(t,\eta) + \eta\mathbf{x}_{f_3}(t,\xi) + (1-\xi)\mathbf{x}_{f_4}(t,\eta) \\ & - (1-\xi)(1-\eta)\mathbf{x}_1(t) - \xi(1-\eta)\mathbf{x}_2(t) - \xi\eta\mathbf{x}_3(t) - (1-\xi)\eta\mathbf{x}_4(t),\end{aligned} \tag{38}$$

where the $\mathbf{x}_i$'s denote coordinates of the corner nodes, the $\mathbf{x}_{f_i}$'s are expressions of the faces, and these terms are all time-dependent for moving grids. If all the faces are represented by one-dimensional iso-parametric mappings of the same order, then the transfinite mapping is equivalent to the iso-parametric mapping in (16). If a face has an exact expression, then that face is represented exactly by the transfinite mapping.

In our case, assume face f_1 is a circular arc (e.g., corresponds to the sliding cell face Ω in Fig. 4), then it can be analytically expressed as

$$\mathbf{x}^{\Omega}(t,\xi) = \begin{bmatrix} x^{\Omega}(t,\xi) \\ y^{\Omega}(t,\xi) \end{bmatrix} = \begin{bmatrix} R\cdot\cos\left[(1-\xi)\theta_1^{\Omega}(t) + \xi\theta_2^{\Omega}(t)\right] + x_c(t) \\ R\cdot\sin\left[(1-\xi)\theta_1^{\Omega}(t) + \xi\theta_2^{\Omega}(t)\right] + y_c(t) \end{bmatrix}, \tag{39}$$

where R and (x_c, y_c) are the radius and the center coordinates of the arc; θ_1^{Ω} and θ_2^{Ω} are the angles that correspond to the starting point (i.e., $\mathbf{x}_1$) and the ending point (i.e., $\mathbf{x}_2$), respectively, of the face. Similarly,

a circular mortar element Ξ_k has the following exact expression,

$$\mathbf{x}^{\Xi_k}(t,z) = \begin{bmatrix} x^{\Xi_k}(t,z) \\ y^{\Xi_k}(t,z) \end{bmatrix} = \begin{bmatrix} R \cdot \cos\left[(1-z)\theta_1^{\Xi_k}(t) + z\theta_2^{\Xi_k}(t)\right] + x_c(t) \\ R \cdot \sin\left[(1-z)\theta_1^{\Xi_k}(t) + z\theta_2^{\Xi_k}(t)\right] + y_c(t) \end{bmatrix}, \tag{40}$$

where $\theta_1^{\Xi_k}$ and $\theta_2^{\Xi_k}$ are the starting and the ending angles of the mortar.

The analytical expressions in (39) and (40) are actually equivalent to the following linear mappings of the angles,

$$\theta^{\Omega} = (1-\xi)\theta_1^{\Omega}(t) + \xi\theta_2^{\Omega}(t), \tag{41}$$

$$\theta^{\Xi_k} = (1-z)\theta_1^{\Xi_k}(t) + z\theta_2^{\Xi_k}(t), \tag{42}$$

where θ^{Ω} is the angle of a physical point (on Ω) that is mapped to a point ξ in the computational space, and θ^{Ξ_k} is the angle of a physical point (on Ξ_k) that is mapped to a point z in the mortar space. Referring to the physical space in Fig. 4 and considering the fact that $\theta^{\Omega} = \theta^{\Xi_k}$ represents the same physical point, the following relation thus holds exactly between the computational space and the mortar space,

$$\xi = o_k + s_k \cdot z, \tag{43}$$

where s_k and o_k are the scaling and the offset of the mortar Ξ_k with respect to the cell face Ω. More specifically,

$$s_k = \frac{\theta_2^{\Xi_k} - \theta_1^{\Xi_k}}{\theta_2^{\Omega} - \theta_1^{\Omega}} = \frac{\Delta\theta^{\Xi_k}}{\Delta\theta^{\Omega}} = \frac{R \cdot \Delta\theta^{\Xi_k}}{R \cdot \Delta\theta^{\Omega}} = \frac{L^{\Xi_k}}{L^{\Omega}}, \tag{44}$$

$$o_k = \frac{\Delta\theta^{\Xi_1} + \Delta\theta^{\Xi_2} + \cdots + \Delta\theta^{\Xi_{k-1}}}{\Delta\theta^{\Omega}} = \sum\nolimits_{\alpha=1}^{k-1} s_\alpha, \tag{45}$$

where L^{Ξ_k} and L^{Ω} are the physical lengths of the mortar and the face, respectively. Note that the scaling and offset are both time-dependent, and are updated when the mortar connectivities are updated. We call the relations in (43)-(45) as the offset and scaling relations, or OS relations for short. For a straight mortar, the OS relations hold exactly, no matter the mortar is represented by a polynomial mortar or transfinite mortar. However, for a curved mortar (more specifically, a circular mortar in this work), the OS relations only hold exactly when the mortar is represented by a transfinite mortar, whereas a polynomial mortar always carries truncation errors.

Two comparison examples of the iso-parametric mapping and the transfinite mapping on elements with a circular edge are included in Appendix A.

4.3. Projection Procedures

Communications on a sliding interface include: projection of local discontinuous values from cell faces to mortars, computation of common values on mortars, and projection of common values back to cell faces. These procedures are discussed in details in what follows. To facilitate the discussion, we adopt the following notations: Q denotes a component of $\mathbf{Q}$; Q_i denotes the discrete value of Q at the i-th FP; $\boldsymbol{Q}$ denotes the vector $(Q_1, Q_2, \cdots, Q_N)$. This same rule also applies to fluxes.

4.3.1. Project local values to mortars

Take solution as an example. Each solution component on a cell face is represented by the following one-dimensional polynomial

$$Q^{\Omega}(\xi) = \sum_{i=1}^{N} Q_i^{\Omega} h_i(\xi). \tag{46}$$

If we define the same set of FPs in the mortar space, then the solution polynomial on a mortar can be constructed in the same way. For example, on the left side of Ξ_k, the solution polynomial is

$$Q^{\Xi_k,\mathrm{L}}(z) = \sum_{i=1}^{N} Q_i^{\Xi_k,\mathrm{L}} h_i(z), \tag{47}$$

where $Q_i^{\Xi_k,\mathrm{L}}$ is a solution component at the i-th FP on the left side of Ξ_k. As illustrated in Fig. 6, to get the solutions on the left side of Ξ_k, we require

$$\int_0^1 \left(Q^{\Xi_k,\mathrm{L}}(z) - Q^{\Omega}(\xi)\right) h_j(z)\ \mathrm{d}z = 0, \quad \forall j = 1, 2, ..., N. \tag{48}$$

Substituting (46) and (47) into the above equation and considering the OS relations, we will get the following equation system

$$\mathbf{M}\boldsymbol{Q}^{\Xi_k,\mathrm{L}} = \mathbf{S}^{\Omega\to\Xi_k}\boldsymbol{Q}^{\Omega}, \tag{49}$$

where the elements of the coefficient matrices are

$$M_{ij} = \int_0^1 h_i(z)h_j(z)\ \mathrm{d}z, \quad i, j = 1, 2, ..., N, \tag{50}$$

$$S_{ij}^{\Omega\to\Xi_k} = \int_0^1 h_i(o_k + s_k z)h_j(z)\ \mathrm{d}z, \quad i, j = 1, 2, ..., N. \tag{51}$$

Solutions of (49), when written in matrix form, are

$$\boldsymbol{Q}^{\Xi_k,\mathrm{L}} = \mathbf{P}^{\Omega\to\Xi_k}\boldsymbol{Q}^{\Omega} = \mathbf{M}^{-1}\mathbf{S}^{\Omega\to\Xi_k}\boldsymbol{Q}^{\Omega}, \tag{52}$$

where $\mathbf{P}^{\Omega\to\Xi_k}$ is the projection matrix from Ω to Ξ_k. This process is repeated for all the solution components.

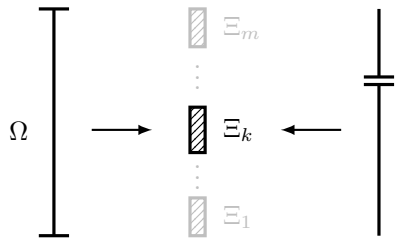


Figure 6: Projection from cell faces to the two sides of a mortar.

In the same way, we can get solutions on the right side of a mortar. Note that the integrals in (50) and (51) can be evaluated exactly and efficiently using quadratures, e.g., the Clenshaw-Curtis quadratures. The Legendre points being the SPs/FPs in this work makes the h_i's orthogonal, and in turn makes the $\mathbf{M}$ matrix diagonal and trivial to invert. The inversion actually needs to be done only once during initialization because the $\mathbf{M}$ matrix is time-independent.

4.3.2. Compute common values on mortars

The common solution $\boldsymbol{Q}^{\Xi_k}$ and the inviscid normal flux $\boldsymbol{F}_{\mathrm{inv}}^{\Xi_k}$ on a mortar are computed in the same way as on a cell interface, i.e.,

$$\boldsymbol{Q}^{\Xi_k} = \frac{1}{2}(\boldsymbol{Q}^{\Xi_k,\mathrm{L}} + \boldsymbol{Q}^{\Xi_k,\mathrm{R}}), \tag{53}$$

$$\boldsymbol{F}_{\mathrm{inv}}^{\Xi_k} = \frac{1}{2}\left[(\overleftrightarrow{\boldsymbol{F}}_{\mathrm{inv}}^{\Xi_k,\mathrm{L}} + \overleftrightarrow{\boldsymbol{F}}_{\mathrm{inv}}^{\Xi_k,\mathrm{R}}) \cdot \mathbf{n} - \lambda(\boldsymbol{Q}^{\Xi_k,\mathrm{R}} - \boldsymbol{Q}^{\Xi_k,\mathrm{L}})\right] - (\mathbf{v}_g \cdot \mathbf{n})\boldsymbol{Q}^{\Xi_k}, \tag{54}$$

where the variables, without further explanation, have similar meanings to those in (22) and (23).

4.3.3. Project common values back to cell faces

Figure 7 illustrates the process of projecting common values from m mortars back to a cell face Ω. Taking flux as an example, we can either directly project back the physical flux (method 1) or convert it to computational flux to project back (method 2). The details are described in what follows.

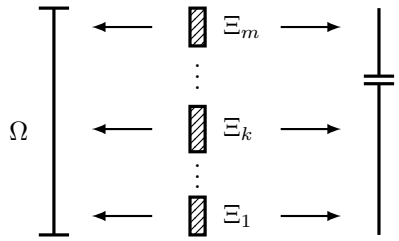


Figure 7: Project common values from mortars back to cell faces.

Method 1:

The inviscid normal fluxes on a cell face and a mortar are represented by the following polynomials,

$$F^{\Omega}_{\text{inv}}(\xi) = \sum_{i=1}^{N} F^{\Omega}_{\text{inv},i} h_i(\xi), \quad F^{\Xi_k}_{\text{inv}}(z) = \sum_{i=1}^{N} F^{\Xi_k}_{\text{inv},i} h_i(z). \tag{55}$$

To get the $F^{\Omega}_{\text{inv},i}$'s, we require

$$\sum_{k=1}^{m} \int_{o_k}^{o_k+s_k} \left(F^{\Omega}_{\text{inv}}(\xi) - F^{\Xi_k}_{\text{inv}}(z) \right) h_j(\xi) \, \mathrm{d}\xi = 0, \quad \forall j = 1, 2, ..., N, \tag{56}$$

which gives

$$\mathbf{M} \boldsymbol{F}^{\Omega}_{\text{inv}} = \sum_{k=1}^{m} s_k \mathbf{S}^{\Xi_k \to \Omega} \boldsymbol{F}^{\Xi_k}_{\text{inv}}, \tag{57}$$

where $\mathbf{M}$ is identical to that in (49), and $\mathbf{S}^{\Xi_k \to \Omega}$ is simply the transpose of the $\mathbf{S}^{\Omega \to \Xi_k}$ matrix from (49). Solutions of (57) are

$$\boldsymbol{F}^{\Omega}_{\text{inv}} = \sum_{k=1}^{m} \mathbf{P}^{\Xi_k \to \Omega} \boldsymbol{F}^{\Xi_k}_{\text{inv}} = \sum_{k=1}^{m} s_k \mathbf{M}^{-1} \mathbf{S}^{\Xi_k \to \Omega} \boldsymbol{F}^{\Xi_k}_{\text{inv}}, \tag{58}$$

where $\mathbf{P}^{\Xi_k \to \Omega}$ is the projection matrix from Ξ_k to Ω. The above physical flux on a cell face is then converted to a computational one to compute residuals. The conversion follows (26), and here the normal is

$$\mathbf{N} = (y^{\Omega}_{\xi}, -x^{\Omega}_{\xi}) = L^{\Omega}(\cos\theta^{\Omega}, \sin\theta^{\Omega}). \tag{59}$$

The final computational flux on the cell face is

$$\widetilde{\boldsymbol{F}}^{\Omega}_{\text{inv}} = \|\mathbf{N}\| \boldsymbol{F}^{\Omega}_{\text{inv}} = L^{\Omega} \boldsymbol{F}^{\Omega}_{\text{inv}}. \tag{60}$$

Method 2:

Alternatively, we can convert the the physical flux in (54) to computational, and then do the back projection. For a mortar, there are two types of normals depending on which space, i.e., the mortar space or the computational space, is taken as the reference space. For the mortar space, the normal is

$$\breve{\mathbf{N}} = (y^{\Xi_k}_{z}, -x^{\Xi_k}_{z}) = L^{\Xi_k}(\cos\theta^{\Xi_k}, \sin\theta^{\Xi_k}). \tag{61}$$

For the computational space, the normal is

$$\widetilde{\mathbf{N}} = (y^{\Xi_k}_{\xi}, -x^{\Xi_k}_{\xi}) = (y^{\Xi_k}_{z} z_{\xi}, -x^{\Xi_k}_{z} z_{\xi}) = \frac{1}{s_k}(y^{\Xi_k}_{z}, -x^{\Xi_k}_{z}) = \frac{1}{s_k}\breve{\mathbf{N}}. \tag{62}$$

The corresponding fluxes in these two spaces are

$$\breve{\boldsymbol{F}}^{\Xi_k}_{\text{inv}} = \|\breve{\mathbf{N}}\| \boldsymbol{F}^{\Xi_k}_{\text{inv}} = L^{\Xi_k} \boldsymbol{F}^{\Xi_k}_{\text{inv}}, \tag{63}$$

$$\widetilde{\boldsymbol{F}}^{\Xi_k}_{\rm inv} = \|\widetilde{\mathbf{N}}\|\boldsymbol{F}^{\Xi_k}_{\rm inv} = \frac{1}{s_k}\breve{\boldsymbol{F}}^{\Xi_k}_{\rm inv}. \tag{64}$$

The computational inviscid fluxes on a cell face and a mortar are represented by the following polynomials,

$$\widetilde{F}^{\Omega}_{\rm inv}(\xi) = \sum_{i=1}^{N} \widetilde{F}^{\Omega}_{{\rm inv},i} h_i(\xi), \quad \widetilde{F}^{\Xi_k}_{\rm inv}(z) = \sum_{i=1}^{N} \widetilde{F}^{\Xi_k}_{{\rm inv},i} h_i(z). \tag{65}$$

To get the $\widetilde{F}^{\Omega}_{{\rm inv},i}$'s, we require

$$\sum_{k=1}^{m} \int_{o_k}^{o_k+s_k} \left(\widetilde{F}^{\Omega}_{\rm inv}(\xi) - \widetilde{F}^{\Xi_k}_{\rm inv}(z) \right) h_j(\xi)\, {\rm d}\xi = 0, \quad \forall j = 1, 2, ..., N, \tag{66}$$

which leads to

$$\mathbf{M}\widetilde{\boldsymbol{F}}^{\Omega}_{\rm inv} = \sum_{k=1}^{m} s_k \mathbf{S}^{\Xi_k\to\Omega} \widetilde{\boldsymbol{F}}^{\Xi_k}_{\rm inv} = \sum_{k=1}^{m} \mathbf{S}^{\Xi_k\to\Omega} \breve{\boldsymbol{F}}^{\Xi_k}_{\rm inv}, \tag{67}$$

where $\mathbf{M}$ and $\mathbf{S}^{\Xi_k\to\Omega}$ are identical to those in (57). The solutions of the above system are

$$\widetilde{\boldsymbol{F}}^{\Omega}_{\rm inv} = \sum_{k=1}^{m} \mathbf{P}^{\Xi_k\to\Omega} \breve{\boldsymbol{F}}^{\Xi_k}_{\rm inv} = \sum_{k=1}^{m} \mathbf{M}^{-1}\mathbf{S}^{\Xi_k\to\Omega} \breve{\boldsymbol{F}}^{\Xi_k}_{\rm inv}, \tag{68}$$

and note the difference between the $\mathbf{P}^{\Xi_k\to\Omega}$ matrix in the above equation and that in Eq. (58).

These two methods are in fact equivalent as one may expect. To show this, simply divide both sides of (68) by L^{Ω}, and consider (44), (60), and (63), and we will get exactly the same result as (58). Also note the singularity in (64) when the scaling (size) of a mortar becomes zero. This singularity is eliminated in (67) where the scaling is multiplied back. Equation (64) is for derivation purpose only, and is not actually evaluated in the computation, therefore the singularity is naturally avoided.

4.3.4. Treatment of viscous fluxes

For viscous flow, we first project the common solution (53) back to cell faces in the same way as (58). The updated solutions on a cell face are then involved in the computation of solution gradients. After that, we could either project local gradients (method 1) or local viscous fluxes (method 2) to mortars to compute common viscous fluxes which are then projected back. The steps are described below.

Method 1:

The local gradients on cell faces are projected to mortars following (52). The common gradients and common physical viscous fluxes on a mortar are then computed following (28) and (27), respectively. The normal viscous flux is calculated as

$$\breve{\boldsymbol{F}}^{\Xi_k}_{\rm vis} = \overleftrightarrow{\boldsymbol{F}}^{\Xi_k}_{\rm vis} \cdot \breve{\mathbf{N}}, \tag{69}$$

where $\overleftrightarrow{\boldsymbol{F}}^{\Xi_k}_{\rm vis} = (\boldsymbol{F}^{\Xi_k}_{\rm vis}, \boldsymbol{G}^{\Xi_k}_{\rm vis})$ with the two components representing the physical common viscous fluxes. This normal viscous flux is finally projected back to cell faces following (68).

Method 2:

Local viscous flux, denoted by $\widetilde{\boldsymbol{F}}^{\Omega}_{\rm vis}$, is projected to mortars in the same way as (48). The resulting normal viscous flux on the left side of a mortar is

$$\breve{\boldsymbol{F}}^{\Xi_k,{\rm L}}_{\rm vis} = \mathbf{P}^{\Omega\to\Xi_k} \widetilde{\boldsymbol{F}}^{\Omega}_{\rm vis} = s_k \mathbf{M}^{-1}\mathbf{S}^{\Omega\to\Xi_k} \widetilde{\boldsymbol{F}}^{\Omega}_{\rm vis}. \tag{70}$$

Note the difference on the $\mathbf{P}^{\Omega\to\Xi_k}$'s in the above equation and that in (52). The viscous flux on the right side of a mortar is obtained in the same way. The common normal viscous flux is then calculated as

$$\breve{\boldsymbol{F}}^{\Xi_k}_{\rm vis} = \frac{1}{2}(\breve{\boldsymbol{F}}^{\Xi_k,{\rm L}}_{\rm vis} + \breve{\boldsymbol{F}}^{\Xi_k,{\rm R}}_{\rm vis}), \tag{71}$$

which is projected back to cell faces following (68) to replace the original normal viscous flux.

We have compared method 1 and method 2 in a series of tests (not reported here), and do not notice any obvious difference on the results. However, method 2 is slightly faster than method 1, because it requires fewer projections and calculations.

The above projection procedures are conservative and retain flow characteristics (i.e., satisfy outflow condition) for linear and circular transfinite mortar elements. The proofs are given in Appendices B and C.

4.4. On the Implementation

A flow chart of the implementation procedures is shown in Fig. 8. Some of the steps are explained with more details in what follows.

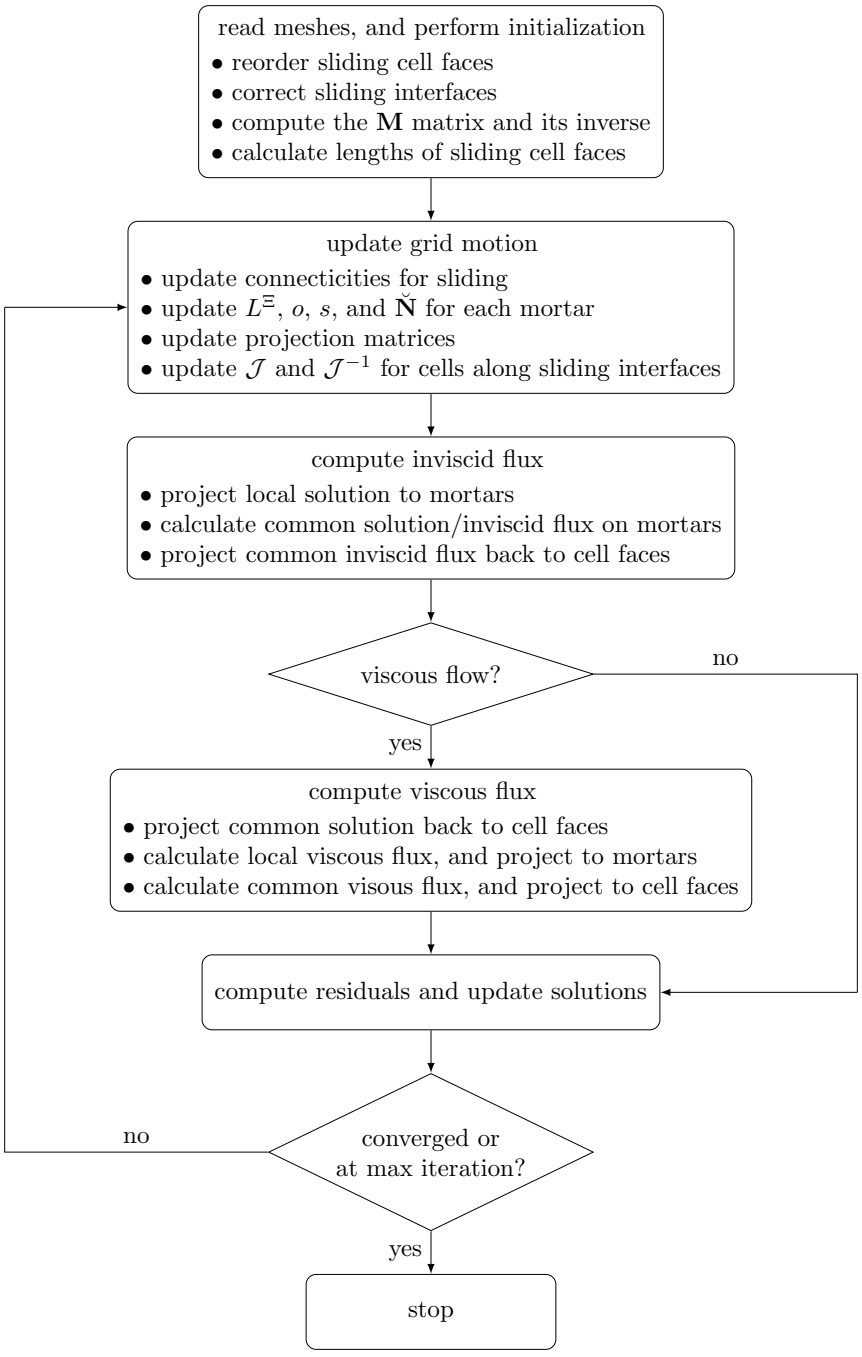


Figure 8: Flow chart of the implementation procedures.

4.4.1. Read meshes

Mesh for each subdomain is generated independently and stored in a separate file. When subdomain meshes are read in, they are assembled into a "single" mesh by adding offsets to the numbering of cells and vertices. For example, as shown in Fig. 9, assume we have n subdomains with N_1^e, N_1^e, N_2^e, ..., N_n^e cell elements. In the assembled mesh, numbering for the cells in subdomain 1 starts from 1, for subdomain 2 starts from $N_1^e + 1$, for subdomain 3 starts from $N_1^e + N_2^e + 1$, and so on. The same rule applies to vertices and boundary faces. In this way, cells and vertices are numbered uniquely and continuously. The overall mesh behaves just like a single mesh, and sliding interfaces are like interior boundaries.

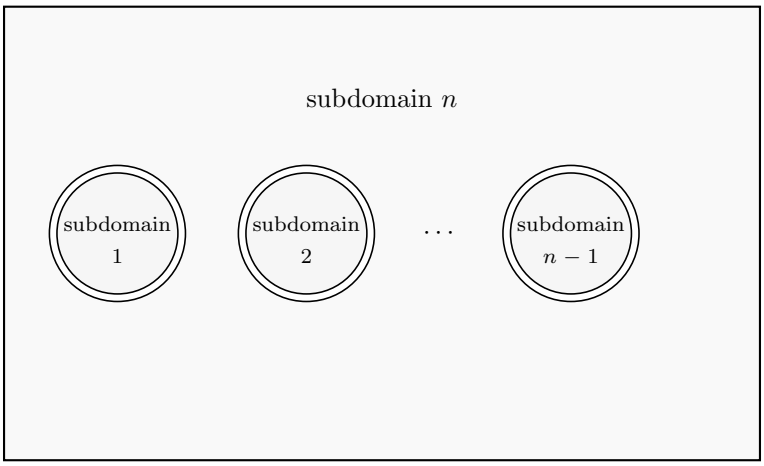


Figure 9: Schematic of n subdomains with $(n-1)$ sliding interfaces (inner subdomains have been scaled).

4.4.2. Reorder sliding cell faces

All sliding cell faces are stored consecutively in an array. This means that cell faces from the left side of the first sliding interface are stored first, followed by those from the right side of the first sliding interface, and then those from the left side of the second sliding interface, and so on. Cell faces from each side are reordered into counterclockwise order, which makes connectivity updating more efficient. The algorithm for the reordering is quite simple: any cell face can be assigned as the first cell face, and then the next cell face is the one whose starting vertex is the ending vortex of the current cell face, repeat until finished.

4.4.3. Correct sliding interface

Mesh generators may introduce geometric errors. When it comes to sliding mesh, the problem is usually that, the vertices on a circular sliding interface do not represent the same sliding radius. This geometric error may potentially contaminate a simulation. Here is an easily fix that can be performed during preprocessing: pick up a vertex and take the corresponding sliding radius as a reference radius, and then use this reference radius and the original angle of each vertex to update the coordinates of that vertex. In this way, all the vertices on a circular sliding interface represent the same sliding radius.

4.4.4. Update connectivities

For simplicity, we only consider the case with one sliding interface, such as that shown in Fig. 3(c). Once the mesh is given, the total number of mortar elements is also given, which is equal to the total number of cell faces on the two sides of a sliding interface. This information allows us to pre-allocate the necessary memories. We then take the starting vertex of the first cell face as the starting vertex of the first mortar, and this connectivity will not change with time. After that, we "walk" along the sliding interface counterclockwise, and the next vertex (no matter which side it is from) becomes the ending vertex of the first mortar and the starting vertex of the second mortar at the same time. We repeat this process until all mortars are identified. This overall process can be applied to arbitrary number of sliding interfaces. More detailed steps can be found in our previous work [33].

4.4.5. Compute fluxes

The flow chart in Fig. 8 only shows the steps related to sliding. These steps must be injected into the main solver at the right locations. For example, the "compute inviscid flux" part is called after interior and cell interface inviscid flux computations are done. The back projection of common solution takes place after all inviscid flux computations are finished and before viscous flux computations start. The projections of viscous flux are carried out after all interior and cell interface viscous flux computations are done.

4.5. Extension to 3D

The above procedures can be readily extended to three-dimensional. We take a simplified mesh (with a cutout to expose the sliding interface) for a rotating square cylinder as shown in Fig. 10(a) to explain how it works. For simplicity, we require the two subdomain meshes to match in the spanwise (i.e., axial) direction. In practice this requirement can be easily lifted. The resulting sliding interface is cylindrical. We take out a

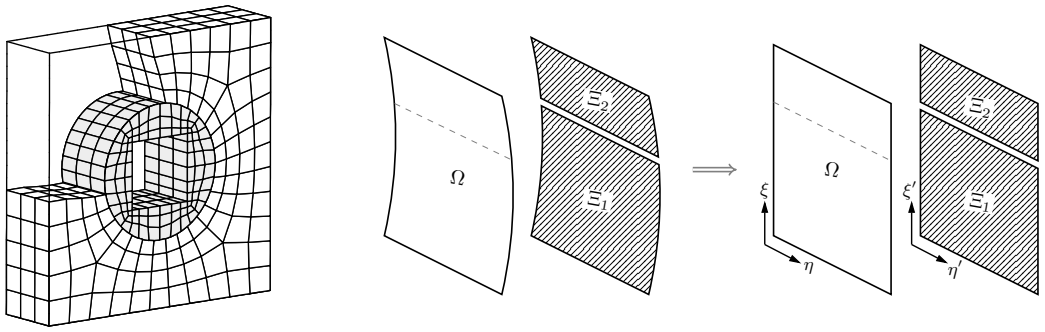


Figure 10: Schematics of: (a) a 3D sliding mesh (with a cutout), (b) mapping of cell face and mortars.

cell face and its mortars (assume two here) as sketched in Fig. 10(b) to demonstrate the process. Transfinite mapping is first applied to map these physical elements to standard elements. The resulting mortar space (ξ', η') and the computational space (ξ, η) are related as

$$\xi = o + s\xi', \quad \eta = \eta', \tag{72}$$

where $0 \leq \xi, \eta, \xi', \eta' \leq 1$, and o and s are the offset and scaling of a mortar. Assume the variable of interest is ϕ, then on a face and the left side of a mortar, we have

$$\phi^{\Omega}(\xi, \eta) = \sum_{j=1}^{N} \sum_{i=1}^{N} \phi_{ij}^{\Omega} h_i(\xi) h_j(\eta), \tag{73}$$

$$\phi^{\Xi,L}(\xi', \eta') = \sum_{j=1}^{N} \sum_{i=1}^{N} \phi_{ij}^{\Xi,L} h_i(\xi') h_j(\eta'), \tag{74}$$

To project ϕ from a cell face to the left side of a mortar, i.e., to obtain the unknown $\left(\phi_{ij}^{\Xi,L}\right)$'s, we require

$$\int_0^1 \int_0^1 (\phi^{\Xi,L}(\xi', \eta') - \phi^{\Omega}(\xi, \eta)) h_\alpha(\xi') h_\beta(\eta') \mathrm{d}\xi' \mathrm{d}\eta' = 0, \ \forall\, \alpha, \beta = 1, 2, ..., N. \tag{75}$$

Substituting (72) into the above equation, it is provable that the projection reduces to the following one-dimensional one,

$$\int_0^1 (\phi^{\Xi,L}(\xi', X_j) - \phi^{\Omega}(\xi, X_j)) h_\alpha(\xi') \mathrm{d}\xi' = 0, \ \forall\, \alpha = 1, 2, ..., N, \tag{76}$$

where X_j is the coordinate of the j-th SP. The above projection is basically identical to (48). We just need to repeat this process for every j in order to obtain all the unknowns. The procedures for calculating the common values and projection them back are also identical to those discussed in Sec. 4.3.

5. Examples

In this section, we apply the sliding mesh method to several flow problems. We first test it on an inviscid flow and a viscous flow to verify the spatial and temporal accuracies. We then study the conservation and the free-stream-preservation properties of the method. Following that, we perform a comparison study between the present method and a rigid-rotation method on flow over a rotating square cylinder. Finally, we apply the method to simulate flow over multiple rotating square cylinders in both 2D and 3D to further demonstrate its capability.

For all the test cases wherever applicable, we employ the following L_2 norm to measure the errors,

$$L_2 \text{ error} = \sqrt{\frac{\sum_{i=1}^{N_{\text{DOF}}} (\phi_i - \phi_i^{\text{exact}})^2}{N_{\text{DOF}}}}, \tag{77}$$

where ϕ represents the variable of interest, ϕ_i and ϕ_i^{exact} are the numerical and the exact solutions at the i-th degree of freedom (DOF), $N_{\text{DOF}} = N_{\text{elem}} \cdot N^2$ is the total number of DOFs, N_{elem} is the total number of mesh elements, $N = P + 1$ is the scheme order (also the number of SPs in each direction of an element), and P is the polynomial degree. Unless otherwise noted, transfinite mortars are used by default.

5.1. Spatial Accuracy

5.1.1. Euler vortex flow

This is an inviscid flow test. In this flow, an isentropic vortex is superimposed to and convected by a uniform flow. The flow field in an infinite domain at a time instant t can be analytically expressed as

$$u = U_\infty \left\{ \cos\theta - \frac{\epsilon y_r}{r_c} \exp\left(\frac{1 - x_r^2 - y_r^2}{2 r_c^2} \right) \right\}, \tag{78}$$

$$v = U_\infty \left\{ \sin\theta + \frac{\epsilon x_r}{r_c} \exp\left(\frac{1 - x_r^2 - y_r^2}{2 r_c^2} \right) \right\}, \tag{79}$$

$$\rho = \rho_\infty \left\{ 1 - \frac{(\gamma - 1)(\epsilon M_\infty)^2}{2} \exp\left(\frac{1 - x_r^2 - y_r^2}{r_c^2} \right) \right\}^{\frac{1}{\gamma - 1}}, \tag{80}$$

$$p = p_\infty \left\{ 1 - \frac{(\gamma - 1)(\epsilon M_\infty)^2}{2} \exp\left(\frac{1 - x_r^2 - y_r^2}{r_c^2} \right) \right\}^{\frac{\gamma}{\gamma - 1}}, \tag{81}$$

where U_∞, ρ_∞, p_∞, M_∞ are the speed, density, pressure and Mach number of the uniform flow; θ denotes the mean flow direction; ϵ and r_c are the strength and size of the vortex; $(x_r, y_r) = (x - x_0 - \bar{u}t, y - y_0 - \bar{v}t)$ are the relative coordinates; (x_0, y_0) represent the initial position of the vortex; $(\bar{u}, \bar{v}) = (U_\infty \cos\theta, U_\infty \sin\theta)$ are the velocity components of the mean flow.

For this test, we have chosen the following parameters: $U_\infty = 1$, $\rho_\infty = 1$, $M_\infty = 0.3$, $\theta = \arctan(1/2)$, $\epsilon = 1$, and $r_c = 1$. The overall computational domain has a size of $0 \le x, y \le 10$, and the vortex is initially

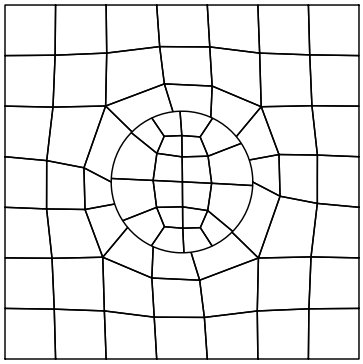
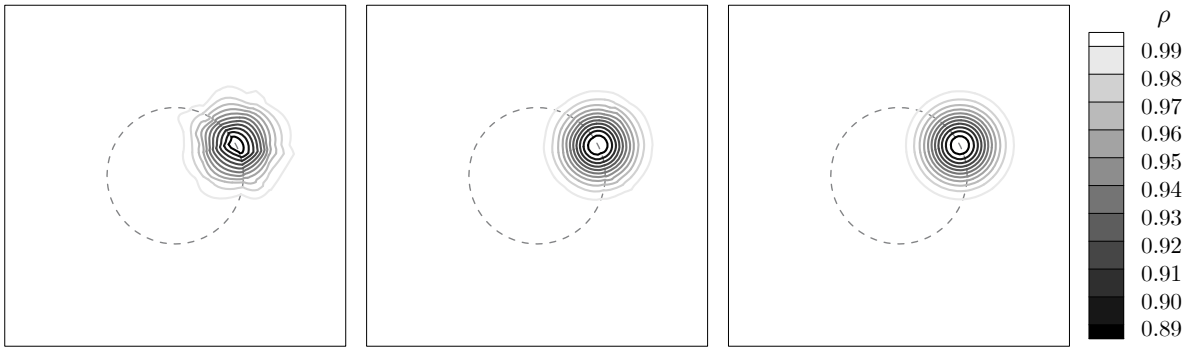


Figure 11: Mesh for Euler vortex flow simulation.

placed at $(x_0, y_0) = (5, 5)$. Time-dependent analytical solutions are weakly prescribed along the boundaries to provide Dirichlet boundary conditions. The mesh used for this simulation is shown in Fig. 11, where there are 72 mesh elements in total, with 20 of them in the inner rotating subdomain whose radius is 2. Six rotational/sliding speeds: $\omega = 0, 1, 5, 10$, 15 and 20, are tested to study their effects on the solution. For all the cases, a five-stage fourth-order SSP Runge-Kutta scheme [41, 42] with a time step size of $\Delta t = 1.0 \times 10^{-4}$ is used for the time marching.

In Fig. 12, we compare the density contours at $t = 2$ from the $\omega = 20$ case using different schemes. At this time instant, the vortex center travels right onto the sliding interface. It is obvious that $P = 2$ does not provide enough resolution as the vortex is poorly resolved. But as the polynomial degree increases, the solution quality becomes much improved. Even at a small polynomial degree of $P = 4$ and on such a coarse mesh, the details of the vortex are very well captured.



(a) $P = 2$ (b) $P = 3$ (c) $P = 4$

Figure 12: Density contours of Euler vortex flow at $t = 2$ (dashed lines represent sliding interface).

The L_2 errors of density are plotted in Fig. 13 against polynomial degrees. It is seen that the errors from different cases are comparable when $P \leq 13$, and start differing when $P > 13$. The reason for this is that spatial errors dominate in the first regime, and they keep decreasing to such a small level that temporal errors start to dominate in the second regime. And larger rotational speed induces larger temporal errors as is indicated by the second regime of Fig. 13. Similar observation on the rotational-speed effects was reported in [25]. In that work, the effects start showing up at a very early stage of $P = 4$, which is mainly

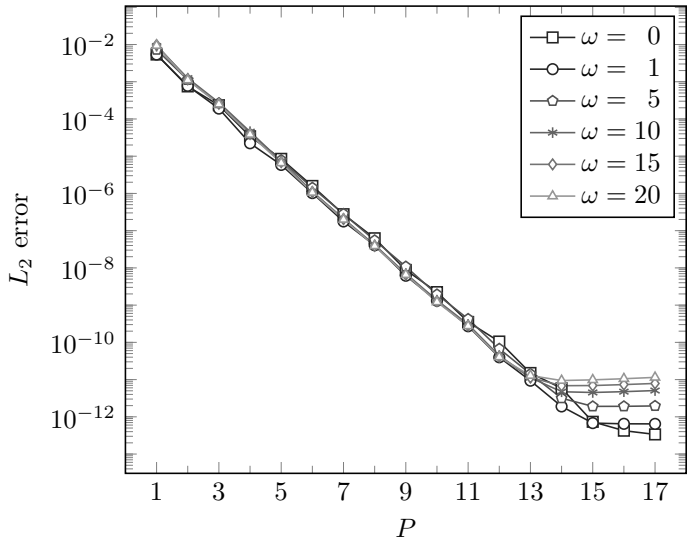


Figure 13: L_2 error of density against polynomial degree for Euler-vortex flow simulations using transfinite mortars.

due to the low-order (2nd order) temporal scheme used in that work. The present results suggest that the rotational/sliding speed effects can be dramatically reduced by using a high-order temporal scheme.

A closer look at the curves in Fig. 13 reveals "inconsistencies" on the results. For example, the $\omega = 20$ case generally gives smaller errors than the $\omega = 5$ case does in the first regime, and the errors are sometimes even smaller than those of the $\omega = 0$ case. These inconsistencies originate from the nonuniformity of the mesh in the rotating subdomain. When the vortex travels to the same location, it is actually captured by different grid resolutions when the rotational speed differs. Nevertheless, for all the cases, the errors inevitably decrease exponentially, which confirms the high-order accuracy of the sliding-mesh method.

As a comparison, we also perform this same test using polynomial mortars. Table 1 shows the results, where "poly." and "trans." stand for polynomial and transfinite mortar, respectively. For each case, the polynomial mortars are represented by polynomials of the same degree as that of the scheme. It is seen that the errors from the polynomial mortars overall approach those from the transfinite mortars as P increases. More specifically, when $P \leq 5$ (which is the most practical region for real applications), the polynomial mortars generate larger errors; when $P \geq 6$, the errors from the two mortar types become indistinguishable. One way to reduce the errors from polynomial mortars is to use higher-degree polynomials for the mortars (similar to what we did in our previous work [30]). For example, use $P+1$ or even higher degree polynomial mortars for a degree P scheme. In spite of that, transfinite mortar is still the better choice because it always guarantees the minimum errors.

ω	mortar	$P=1$	2	3	4	5	6	7	8
0	poly.	2.3285E-2	2.9779E-3	2.8862E-4	3.3855E-5	8.5224E-6	1.5871E-6	2.7668E-7	6.2455E-8
	trans.	5.3911E-3	7.5032E-4	2.3443E-4	3.4261E-5	8.4903E-6	1.5878E-6	2.7667E-7	6.2454E-8
1	poly.	1.8387E-2	2.5269E-3	2.3167E-4	2.2928E-5	5.8466E-6	1.0217E-6	1.7674E-7	4.0591E-8
	trans.	5.3807E-3	7.7627E-4	1.9015E-4	2.2283E-5	5.8395E-6	1.0218E-6	1.7674E-7	4.0591E-8
5	poly.	3.7218E-2	2.1340E-3	3.3263E-4	3.9312E-5	7.8944E-6	1.4231E-6	2.8225E-7	5.5027E-8
	trans.	7.2559E-3	1.1192E-3	2.6537E-4	3.9083E-5	7.8713E-6	1.4233E-6	2.8224E-7	5.5028E-8
10	poly.	5.2106E-2	1.8785E-3	3.3372E-4	4.5660E-5	7.4160E-6	1.1199E-6	2.1073E-7	4.1293E-8
	trans.	9.0595E-3	1.2478E-3	2.7582E-4	4.5573E-5	7.3942E-6	1.1201E-6	2.1072E-7	4.1293E-8
15	poly.	6.6791E-2	1.7160E-3	2.9753E-4	3.9792E-5	6.9517E-6	1.0915E-6	2.0459E-7	3.9185E-8
	trans.	9.6755E-3	1.1657E-3	2.5797E-4	3.9764E-5	6.9349E-6	1.0917E-6	2.0459E-7	3.9185E-8
20	poly.	8.3303E-2	1.6067E-3	2.9449E-4	3.8528E-5	6.6154E-6	1.0620E-6	2.0005E-7	3.9062E-8
	trans.	9.3835E-3	1.1098E-3	2.4931E-4	3.8419E-5	6.5872E-6	1.0620E-6	2.0006E-7	3.9062E-8

Table 1: Comparison of the L_2 errors of density for Euler vortex flow simulations using polynomial and transfinite mortars.

5.1.2. Taylor-Couette flow

Taylor-Couette flow is formed between two concentric rotating circular cylinders. Due to viscous effects, this flow will reach a steady state if the Reynolds number is small. The steady-state azimuthal flow speed has the following expression,

$$v_\theta = v_{\theta_i} \frac{r_o/r - r/r_o}{r_o/r_i - r_i/r_o} + v_{\theta_o} \frac{r/r_i - r_i/r}{r_o/r_i - r_i/r_o}, \tag{82}$$

where r_i and r_o are the radii of the inner and the outer boundaries, respectively; v_{θ_i} and v_{θ_o} are the azimuthal flow speeds at these two boundaries.

The mesh for this simulation is shown in Fig. 14. The overall domain is bounded by $r_i = 1$ and $r_o = 2$, and is split into a rotating inner subdomain and a fixed outer subdomain by a sliding interface at $r_s = 1.5$. These two subdomains are meshed into 24 and 32 elements, with uniform grid distribution in azimuthal and radial directions. The outer boundary is treated as a no-slip isothermal wall with $v_{\theta_o} = 0$. The inner boundary is set as a Dirichlet boundary with $\rho_i = 1$, Mach number $Ma = 0.1$, and $v_{\theta_i} = 1$. The Reynolds number based on flow properties at the inner boundary is $Re = 10$. Again, six rotational speeds: $\omega = 0, 1, 5, 10,$ 15 and 20, are tested. The same time marching scheme as that for the previous test with a time step size of $\Delta t = 5.0 \times 10^{-5}$ is used for all the cases. Exact transfinite mapping is employed on all circular boundaries, and all interior cell faces are represented linearly.

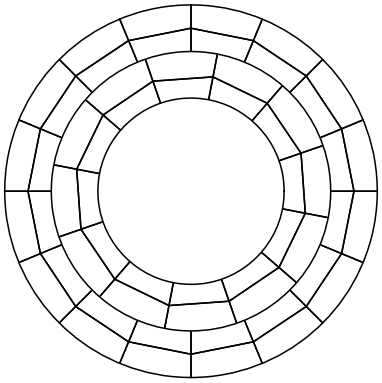
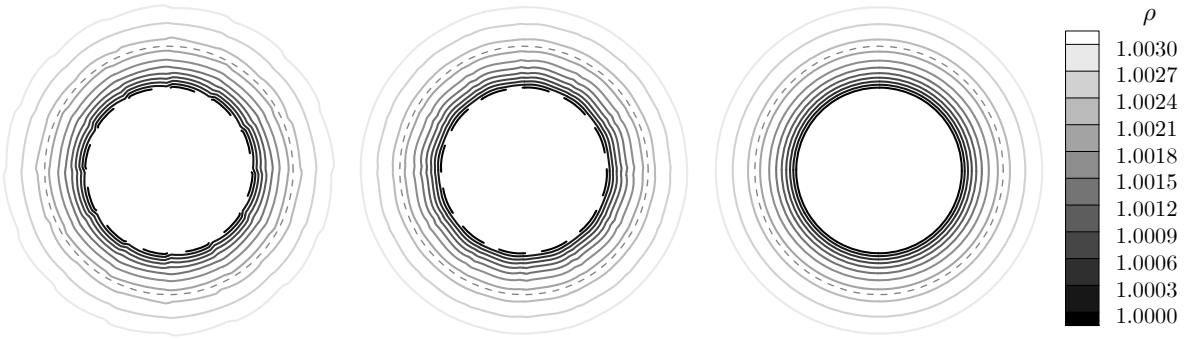


Figure 14: Mesh for Taylor-Couette flow simulation.

Figure 15 shows the steady state density contours from different schemes. The contours are expected to be a series of concentric circles, and we see improved results as the polynomial degree increases. Note that since the boundary conditions are weakly imposed, the computed values on the inner boundary are therefore not necessarily identical to the prescribed values. And this is the reason for the dashed lines at the inner boundaries. Nevertheless, the boundary conditions becomes stricter as the polynomial degree increases, which is why the dashed lines have disappeared at $P = 4$.



(a) $P = 2$ (b) $P = 3$ (c) $P = 4$

Figure 15: Steady state density contours of Taylor-Couette flow (dashed lines in the middle represent sliding interface).

The L_2 errors of the u velocity (i.e., the x-component of v_θ) are shown in Fig. 16. Overall, the errors decrease exponentially with polynomial degree before temporal errors become dominant, which confirms the high-order accuracy of the present method on viscous flow. Meanwhile, the results from different rotational speeds are more consistent than those in the previous test. This is because of the uniformity of the current mesh in the azimuthal direction, which gives identical mesh resolution even when the rotational speed differs. As expected, the rotational speed effects again show up at large P when spatial errors are small enough. We also performed this test using polynomial mortars, and the differences against transfinite mortars are similar to those in the previous test. For this reason, the data is not reported here.

5.2. Temporal Accuracy

In this section, we test the temporal accuracy and verify that the transfinite-mortar based sliding-mesh method does not affect the order of accuracy of a time marching scheme. We employ several strong-stability-preserving Runge-Kutta (SSPRK) schemes for the temporal discretization. Following the rules in [48], we denote an s-stage p-th order SSPRK scheme as SSP(s, p). The CFL condition requires small enough time step size to stabilize a scheme, which makes it difficult for temporal error to dominate. To alleviate this issue, we employ the following SSPRK schemes that allow larger CFL numbers (and thus larger time step sizes): the SSP(4,2) scheme from [48], the SSP(8,3) scheme from [42], and the SSP(10,4) scheme from [49].

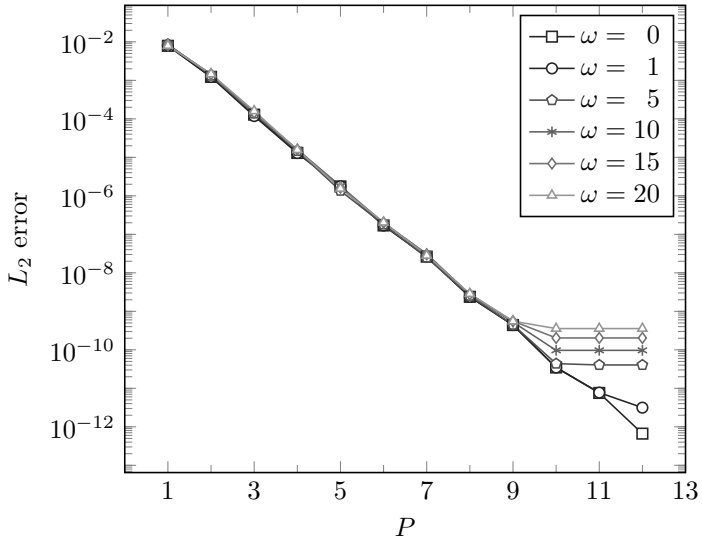


Figure 16: L_2 errors of u velocity against polynomial degree for Taylor-Couette flow simulation.

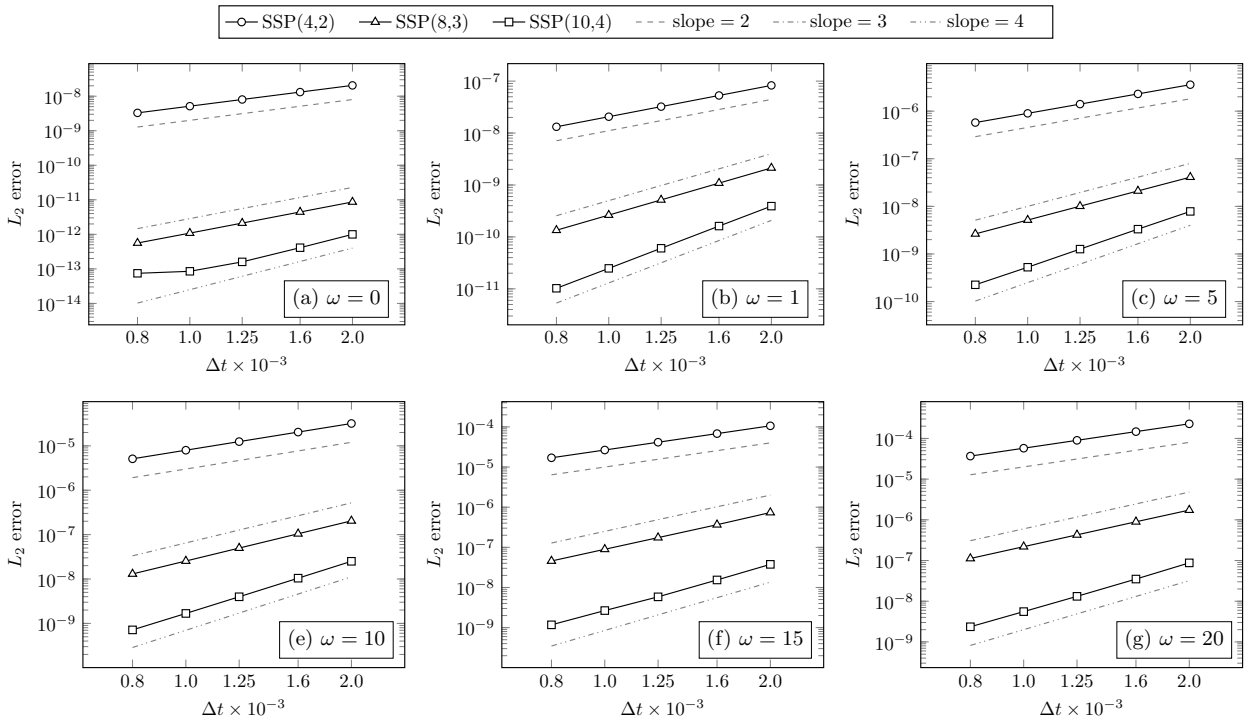


Figure 17: L_2 errors of ρ against time step size for the Euler vortex flow simulation.

The tests are performed on the previous Euler vortex flow. Whenever it permits, we have varied the polynomial degree for each case to ensure that the spatial errors are negligibly small compared with the temporal errors. The results are reported in Fig. 17 for different rotational speeds. For almost all the cases, the temporal errors decrease at the correct orders as the time step size decreases. For the $\omega = 0$ case with SSP(10,4), the temporal errors become so small (in the order of 10^{-14}) that they can no longer be easily separated from the spatial errors at small time step sizes, which results in the incorrect slope at small time step sizes. Nevertheless, at large time step sizes, the curve still shows the correct slope for this case. We also have tested the temporal accuracy on the Taylor-Couette flow, and similar results were obtained (for

conciseness, the results are not included here).

5.3. Conservation

From the conservation proof in Appendix B, and assume a mesh has N_{elem} elements, N_{boun1} boundary cell faces that are mapped to the ξ direction, and N_{boun2} boundary cell faces that are mapped to the η direction, then global conservation can be expressed as

$$\sum_{e=1}^{N_{\text{elem}}} \int_0^1 \int_0^1 \frac{\partial \widetilde{\mathbf{Q}}^e}{\partial t} \mathrm{d}\xi \mathrm{d}\eta + \sum_{b_1=1}^{N_{\text{boun1}}} \int_0^1 \text{sign}^{b_1} \cdot \widehat{\mathbf{F}}^{b_1} \mathrm{d}\eta + \sum_{b_2=1}^{N_{\text{boun2}}} \int_0^1 \text{sign}^{b_2} \cdot \widehat{\mathbf{G}}^{b_2} \mathrm{d}\xi = \mathbf{0}, \tag{83}$$

where sign $= 1$ or -1, depending on which standard face the physical cell face is mapped to. Denoting the temporal discretization by $\widetilde{\partial}/\widetilde{\partial} t$, then the exact time derivative in (83) can be expressed as

$$\frac{\partial \widetilde{\mathbf{Q}}^e}{\partial t} = \frac{\widetilde{\partial} \widetilde{\mathbf{Q}}^e}{\widetilde{\partial} t} + \boldsymbol{\epsilon}(\Delta t, \xi, \eta), \tag{84}$$

where $\boldsymbol{\epsilon}(\Delta t, \xi, \eta)$ represents a small error that is a function of time-step size and space. Substitute the above relation into (83),

$$\sum_{e=1}^{N_{\text{elem}}} \int_0^1 \int_0^1 \frac{\widetilde{\partial} \widetilde{\mathbf{Q}}^e}{\widetilde{\partial} t} \mathrm{d}\xi \mathrm{d}\eta + \sum_{b_1=1}^{N_{\text{boun1}}} \int_0^1 \text{sign}^{b_1} \cdot \widehat{\mathbf{F}}^{b_1} \mathrm{d}\eta + \sum_{b_2=1}^{N_{\text{boun2}}} \int_0^1 \text{sign}^{b_2} \cdot \widehat{\mathbf{G}}^{b_2} \mathrm{d}\xi = \mathbf{E}(\Delta t), \tag{85}$$

where

$$\mathbf{E}(\Delta t) = -\sum_{e=1}^{N_{\text{elem}}} \int_0^1 \int_0^1 \boldsymbol{\epsilon}^e(\Delta t, \xi, \eta) \mathrm{d}\xi \mathrm{d}\eta. \tag{86}$$

When $\Delta t \to 0$, or when it is a steady-state problem, we will have $\mathbf{E}(\Delta t) \to 0$ if a numerical scheme is indeed conservative. And the four components of $\mathbf{E}(\Delta t)$ measure how well conservation is satisfied for mass, momentums and energy.

Based on the above analysis, we employ a flat-plate Couette flow to verify the conservation property of the present method. This flow is formed between two infinite flat plates separated by a distance of H, with the upper plate moving at a constant speed U and the lower plate fixed. The steady state solution is

$$u = U\frac{y}{H}, \quad v = 0, \quad p = \text{constant}, \tag{87}$$

$$T = T_0 + \left(T_1 - T_0\right)\frac{y}{H} + \frac{\mu U^2}{2\kappa}\left(\frac{y}{H} - \frac{y^2}{H^2}\right), \tag{88}$$

where $0 \le y \le H$ is the vertical coordinate, T_0 and T_1 are temperatures on the lower and the upper plates.

The mesh used for this simulation is identical to that in Fig. 11, but is scaled by a factor of $1/10$ in each direction (i.e., with $H = 1$) to make the simulation setup easier. The upper plate speed is set to $U = 1$. The temperatures are set to $T_0 = T_1 = 1$. All other parameters are chosen such that the flow has a Prandtl number $Pr = 0.72$ and Reynolds number $Re = 100$, and the flow on the upper plate has a Mach number $Ma = 0.8$. The flow field is initialized using the exact solution. The boundary conditions are weakly imposed using the exact solution. The SSP(2,2) temporal scheme with a time step size of 5.0×10^{-6} is used for all the simulations. And all the simulations are performed using double precision float numbers.

We have tested different rotational speeds and schemes, with $\mathbf{E}(\Delta t)$ monitored until $Ut/H = 10$. It was observed that for all the cases, $\mathbf{E}(\Delta t)$ slightly oscillates around machine precision, but overall does not grow with time. In the calculations, the numerical time derivative term in (85) is replaced by the residual according to (34), and the integrals are evaluated using Gauss–Legendre quadrature. For conciseness, we only report the time-averaged values of $|\mathbf{E}(\Delta t)|$ for some of the cases in Tab. 2. Obviously, the method based on transfinite mortar has excellent conservation properties even under dramatically different rotational speeds and schemes. This same test was repeated using polynomial mortars, and the results were found to be one to two magnitudes worse when $P \le 5$, but of similar quality when $P \ge 6$. For conciseness, those results are not included here.

ω	P	Mass	Momentum 1	Momentum 2	Energy
0	2	3.457E-16	2.016E-16	3.865E-16	4.284E-16
	4	5.489E-16	3.053E-16	5.120E-16	1.715E-16
	8	5.976E-17	3.939E-16	7.077E-16	7.978E-16
5	2	2.180E-15	9.532E-15	1.532E-15	7.061E-15
	4	7.535E-16	2.850E-16	3.237E-16	5.843E-16
	8	4.442E-16	1.544E-16	3.045E-17	4.136E-16
10	2	6.762E-15	9.032E-15	1.134E-15	1.083E-15
	4	7.486E-16	7.410E-16	5.596E-16	5.025E-16
	8	1.460E-16	1.093E-16	4.794E-16	3.744E-16
20	2	8.135E-15	6.418E-15	3.526E-16	1.253E-15
	4	3.945E-16	3.938E-16	4.947E-16	6.789E-16
	8	8.707E-16	9.576E-16	1.096E-16	4.623E-15

Table 2: Conservation of mass, momentum and energy on a flat-plate Couette flow using transfinite mortars.

5.4. Free-Stream Preservation

In this test, the free stream flow is chosen such that it has a Mach number $Ma = 0.3$. The same mesh as that for the conservation test are used for this test. Dirichlet boundary conditions derived from the constant flow are applied at all the outer boundaries. Different polynomials and rotational speeds are tested to investigate their effects on free-stream preservation. For all the cases, we employ SSP(10,4) with a time-step size of 1.0×10^{-3} as the time marching scheme. We measure the free-stream preservation by the normalized L_2 error of pressure (represents the average strength of numerically generated disturbances). As a comparison, we also test free-stream preservation on a conforming dynamic mesh that has almost exactly the same resolution as the nonconforming sliding mesh. The center of conforming mesh has a displacement of $\Delta y(t) = 0.1 \sin t$ with respect to its initial position, with the mesh in the rest of the domain deforming using an algebraic function [31]. For all the cases, we monitored the L_2 errors until the nondimensional time reaches $U_\infty t/H = 20$. The results at the end of the simulations are summarized in Tabs. 3 and 4.

For perfect free-stream preservation, the error should stay at the level of machine precision, and its magnitude should not change dramatically with polynomial degree. The row marked by "$*$" in Tab. 3 demonstrates that the FR method together with the GCL equations ensure very good free-stream preservation on the conforming dynamic mesh. From the remaining data in both tables, we see that the sliding mesh methods based on both mortar types, however, do not perfectly satisfy free-stream preservation. Instead, they approach free-stream preserving in an exponential manner as the polynomial degree increases. Furthermore, the numerical disturbances introduced by polynomial mortars are consistently about three magnitudes larger than those introduced by transfinite mortars.

ω \ P	1	2	3	4	5	6	7	8
$*$	7.760E-16	6.213E-16	1.983E-15	3.869E-15	4.009E-15	3.566E-15	2.991E-15	2.955E-15
0	1.071E-04	7.844E-06	2.585E-07	8.252E-09	2.173E-10	4.676E-12	1.049E-13	3.332E-15
1	1.023E-04	7.609E-06	2.006E-07	7.495E-09	1.625E-10	3.985E-12	7.057E-14	3.213E-15
5	8.846E-05	7.642E-06	2.075E-07	7.393E-09	1.614E-10	4.006E-12	7.465E-14	3.518E-15
10	8.556E-05	7.559E-06	2.204E-07	7.283E-09	1.691E-10	4.036E-12	7.857E-14	3.364E-15
15	9.179E-05	7.524E-06	2.491E-07	7.806E-09	2.068E-10	4.455E-12	9.981E-14	3.658E-15
20	9.482E-05	7.817E-06	3.456E-07	8.873E-09	2.612E-10	5.123E-12	1.149E-13	3.880E-15

Table 3: Free stream preservation on a conforming dynamic mesh ('$*$') and a sliding mesh (the rest, using transfinite mortars).

ω \ P	1	2	3	4	5	6	7	8
0	2.660E-02	2.745E-03	1.432E-04	5.970E-06	2.786E-07	8.370E-09	3.410E-10	8.074E-12
1	1.773E-02	1.846E-03	8.810E-05	5.914E-06	1.956E-07	6.686E-09	2.273E-10	6.164E-12
5	1.494E-02	2.398E-03	1.087E-04	5.830E-06	1.764E-07	6.651E-09	1.891E-10	5.842E-12
10	1.752E-02	2.923E-03	1.149E-04	4.889E-06	1.557E-07	7.022E-09	1.710E-10	5.647E-12
15	1.955E-02	3.007E-03	1.112E-04	7.051E-06	2.139E-07	1.007E-08	2.215E-10	8.484E-12
20	2.278E-02	2.954E-03	1.383E-04	8.087E-06	2.507E-07	1.263E-08	2.947E-10	8.571E-12

Table 4: Free stream preservation on a sliding mesh using polynomial mortars.

The reason for the present method not strictly satisfying free-stream preservation is not hard to be identified. For a constant free stream flow, we are basically projecting the metric terms between cell faces and mortars. When a circular cell face is represented exactly, its metric terms are in the space of $\mathbf{P}_\infty$, but the output of a projection is in the space of $\mathbf{P}_{N-1}$. This means that there is always a truncation error of $\mathcal{O}(\xi^N)$ during the projection of the metric terms, which therefore results in an exponential approximation of free-stream preservation.

The issue of using exact geometric expression for calculating metric terms has been discussed in [12, 50], and the suggestion is to use polynomial approximation to satisfy free-stream preservation. This suggestion has been proven to work perfectly on curved conforming meshes. A recent work [51] on curved nonconforming meshes in the special scenario of subdivision of parent element reveals that a necessary condition for free-stream preservation is that the metrics of a child face being computed from its parent face. But for general curved nonconforming meshes, such as the sliding meshes in this work, a child face (e.g., a mortar) does not have a unique parent (more specifically, a child face has two parent faces that are different polynomials), and thus the aforementioned necessary condition generally can not be satisfied. To the authors' knowledge, free-stream preservation on general curved nonconforming meshes still remains an open problem for all polynomial-based high-order methods. Note that for linear nonconforming meshes, such as that in [34], free-stream preservation can always be easily satisfied. However, linear mesh obviously introduces too large geometric errors for the curved sliding interfaces here (see Appendix A).

Nevertheless, Tab. 3 shows that, using transfinite mortars, the free-stream-preservation errors from the sliding mesh quickly decreases to machine precision even at moderate polynomial degrees. Since this test is performed on a very coarse grid, we thus expect very minor free-stream preservation effects in real applications where the meshes are usually much finer.

5.5. Flow over a Rotating Square Cylinder

As shown in Fig. 18, we employ two types of meshes for this simulation: a sliding mesh (on the left) and a rigid-rotating mesh (on the right). The square cylinder has a diameter (i.e., diagonal length) of $D = 1$. The sliding mesh consists of 6,797 quadrilateral elements in an overall $100D \times 100D$ domain, with 95 of the elements inside a small rotating subdomain whose diameter is $1.2D$. The rigid-rotating mesh consists of 6,841 elements in a whole-piece rotating circular domain whose diameter is $100D$. Both meshes are refined in the vicinity of the cylinder to better resolve flow structures. With similar amounts of mesh elements, the sliding mesh obviously provides good resolution in a wider range of the wake region. The rigid-rotating mesh, on the other hand, always wastes a large amount of elements in unimportant regions, which is unavoidable. Close views of the meshes are shown on the top right of each figure.

The freestream flow is chosen such that it has a Mach number $Ma = 0.1$. The Reynolds number based on free-stream flow properties and the cylinder diameter is $Re_D = 100$. For the sliding mesh, the inner subdomain rotates counterclockwise at a non-dimensional rotational speed $\omega D/U_\infty = \pi/2$ (which corresponds to a nondimensional period $T^* = 4$); and for the rigid-rotating mesh, the whole domain rotates at this speed. No-slip adiabatic wall boundary condition is applied on the cylinder surface. Characteristic far-field boundary conditions are applied at all outer boundaries. The eighth-order (i.e., $P = 7$) spatial scheme and the SSP(10,4) time marching scheme [49] with a nondimensional time-step size of $\Delta t U_\infty/D = 2.0 \times 10^{-4}$ are employed to run the simulations. Polynomial and mesh refinement studies have also been performed to have confirmed that the present setup well resolves the flow.

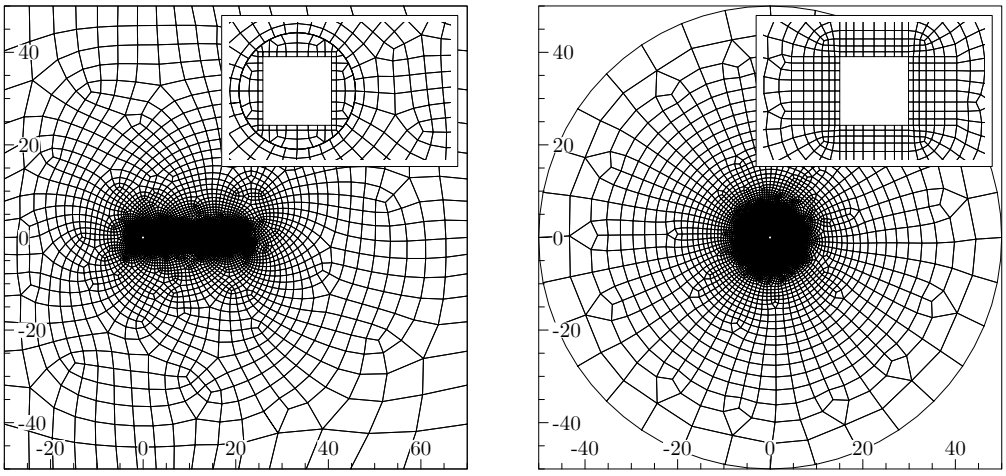


Figure 18: Meshes for a rotating square cylinder: left, sliding mesh; right, rigid-rotating mesh.

The lift and drag coefficients of the cylinder are plotted in Fig. 19. The cylinder experiences a negative lift and positive drag all the time. These curves overall do not have a periodic pattern that is directly related to the cylinder's motion. But the horizontal distance between two neighboring local peaks (or troughs) is approximately 1 (i.e., $T^*/4$), which is the period at which the cylinder disturbs the flow. Results from these two distinctly different approaches agree very well, which confirms the correctness of the present method.

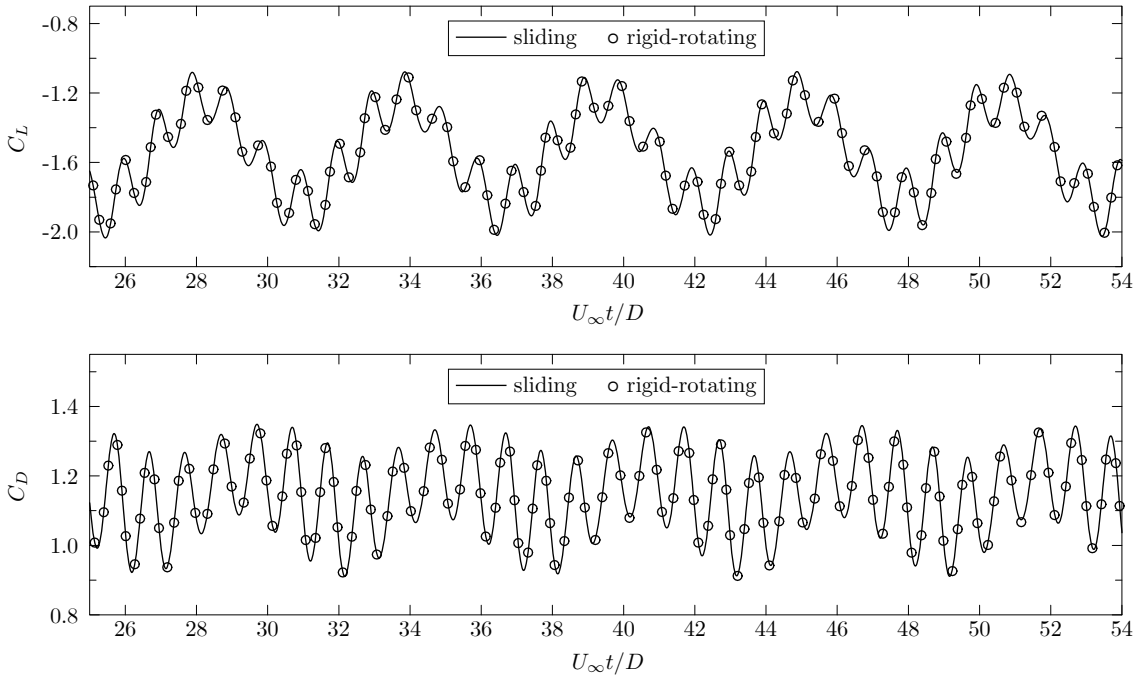


Figure 19: Lift and drag coefficients of a rotating square cylinder using a sliding mesh and a rigid-rotating mesh.

We also compare the flow fields from the same time instant in Fig. 20 using vorticity contours. Alternative positive and negative vortices are observed in the wake region, and they are well captured even in the very far flow region. The two approaches basically produce identical flow fields as the vortices have almost exactly the same positions and shapes. The far wake region from the sliding mesh is slightly better resolved than that from the rigid-rotating mesh. This is consistent with our previous observation that sliding mesh always provides better resolution than rigid-rotating mesh for a given number of elements.

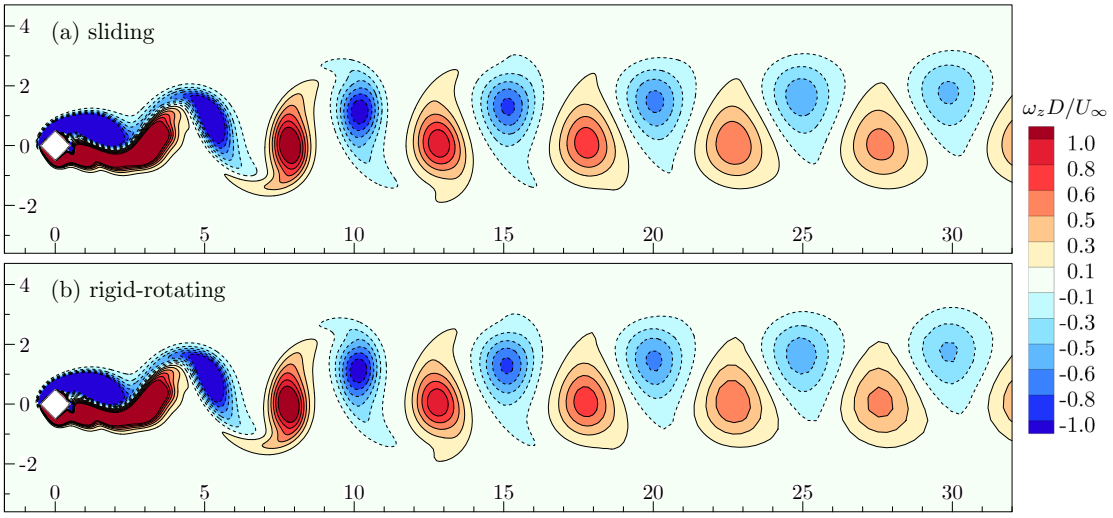
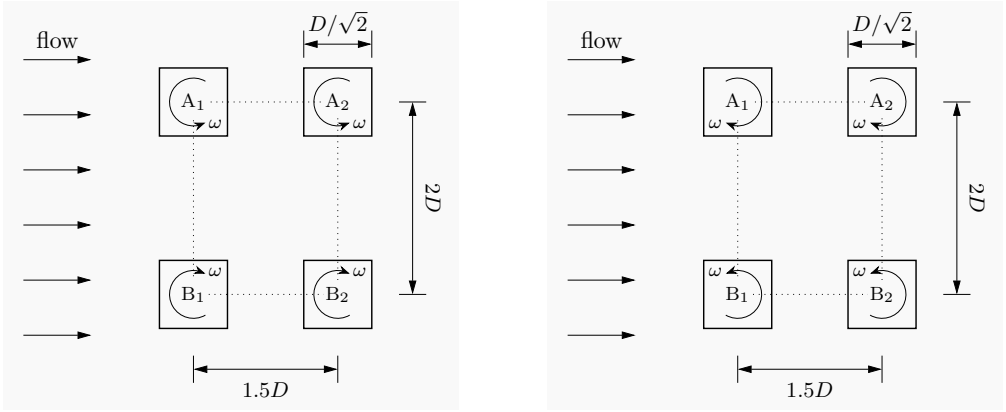


Figure 20: Vorticity contours of flow over a rotating square cylinder at a time instant.

5.6. Flow over Multiple Rotating Square Cylinders

In this test, we simulate the interactions of a uniform incoming flow and four rotating square cylinders to show the capability of the method for handling multiple rotating objects. A schematic of the simulation setup is shown in Fig. 21. The four cylinders are separated by a horizontal distance of $1.5D$ and a vertical distance of $2D$, where D is the diameter of the cylinder with the same definition as that in the previous test. Two cases are investigated, where the only difference is the rotational direction of each cylinder as marked in the schematic. For simplicity, we denote the four cylinders as A_1, A_2, B_1 and B_2.



(a) Case 1 (b) Case 2

Figure 21: Schematic of simulation setup for flow over multiple rotating square cylinders.

The overall mesh for this test has 7,409 elements, with its topology and distribution similar to those of the sliding mesh in the previous test. Each inner subdomain mesh is exactly the same as that in the previous test. For conciseness, views of the mesh are not repeated here. All other parameters for this test, such as the freestream Mach number, the Reynolds number, the rotational speed, the boundary conditions, etc., are exactly the same as those for the previous test. The simulations are performed using the eighth-order scheme with SSP(10,4) and a nondimensional time step size of 1.25×10^{-4}. Each simulation is continued for a nondimensional time of 500. In what follows, we briefly discuss the results for each case.

5.6.1. Case 1

Figure 22 shows the lift and drag coefficients from $U_\infty t/D = 50$ to 100. Due to the Magnus effects and the opposite rotational directions, the cylinders in the same column experience lifts with the same magnitude but opposite signs. The overall system thus does not experience any net lift force. On average, the upstream cylinders have much larger lifts but much smaller lift oscillations than the downstream ones do. Similarly, the cylinders in the same column experience exactly the same drags both in magnitudes and signs in this time frame. The drags on the upstream cylinders also have much larger magnitude but much smaller oscillations than those of the downstream ones. In fact, the magnitudes of the forces on the cylinders in the same column do not stay the same all the time, and they actually start differing after $U_\infty t/D \approx 130$ (not shown in the figure, will explain later).

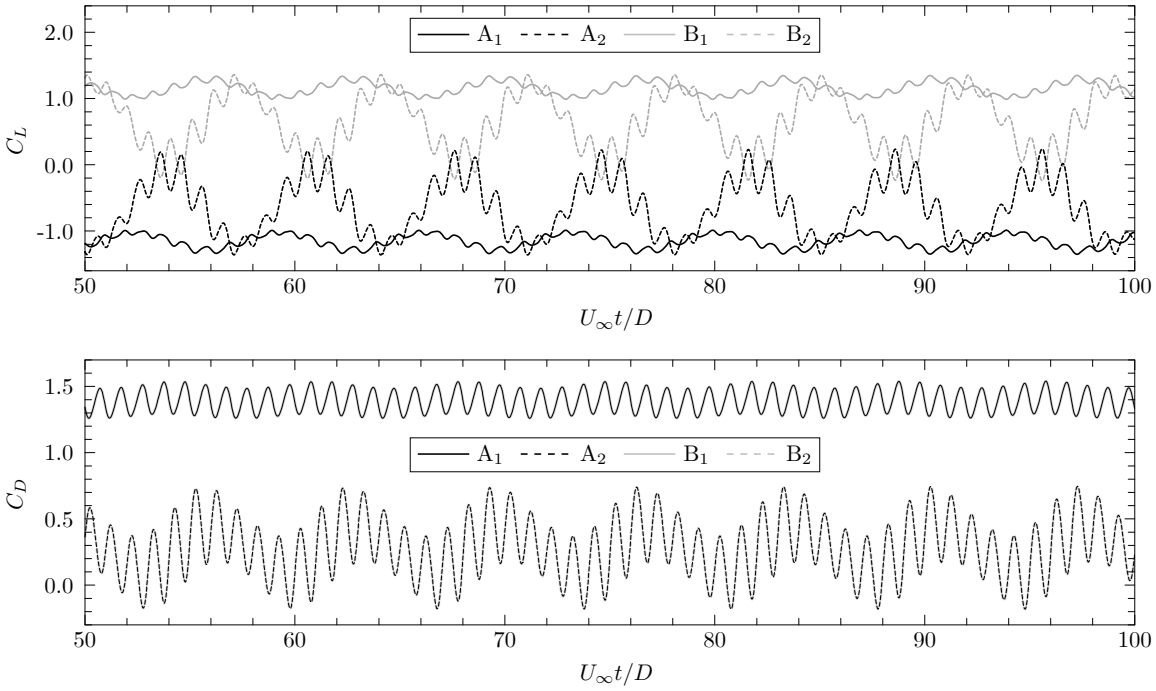


Figure 22: Lift and drag coefficients for flow over multiple rotating square cylinders (Case 1).

A snapshot of the flow field visualized by vorticity contours at $U_\infty t/D = 80.5$ is shown in Fig. 23. Well organized positive-negative vortex pairs are seen in the wake region and extend all the way to the very far flow field. This patten is perfectly mirror-symmetric about the horizontal center line of the setup. In fact, this is a very unstable and unsustainable system. Any tiny numerical disturbance could cause the flow to lose this symmetry. Using the present method, this symmetry is maintained for a surprisingly long time ($U_\infty t/D \approx 130$) before breaking up. This evidently demonstrates that the present method introduces very little numerical disturbance to a simulation. After the breakup of the symmetry, the flow becomes less interesting, and therefore we only focus on the flow before that point for this case.

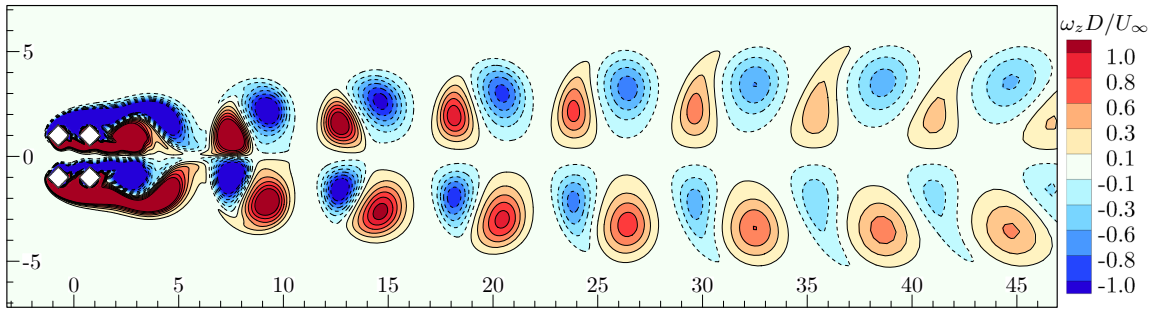


Figure 23: Vorticity contours of flow over multiple rotating square cylinders (Case 1).

5.6.2. Case 2

The lift and drag coefficients for this case are plotted in Fig. 24. The previous conclusions from Case 1 still hold here. Except that the current curves show a rather mono frequency that corresponds to a period of $T^*/4$, where $T^* = 4$ is the nondimensional rotational period of the cylinders. This mono frequency indicates fewer vortical structures (or noises) in the flow field. Compared with Case 1, we also notice dramatic drag reduction on all the cylinders. On average, the system has a drag reduction of about 50%, and the rear cylinders even do not experience much net drag at all. Meanwhile, lift enhancement is observed on each individual cylinder, but the overall system still remains lift-free.

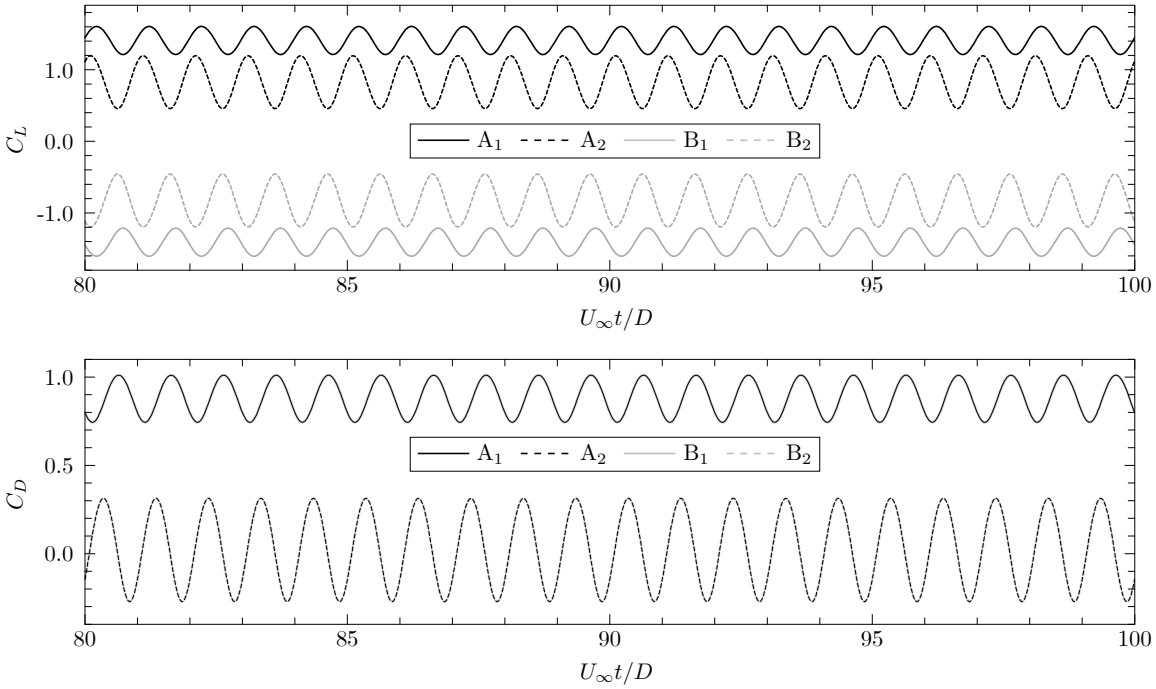


Figure 24: Lift and drag coefficients for flow over multiple rotating square cylinders (Case 2).

Figure 25 shows the instantaneous flow field at $U_\infty t/D = 250.5$. Compared with Case 1, vortex shedding has been completely suppressed in this case, which is consistent with our observation from the force curves. This flow remains very stable and is almost steady (except in the very vicinity of the cylinders) throughout the simulation (i.e., even at $U_\infty t/D = 500$). The reason for this steadiness is obvious. The rotation motion of the cylinders in this case (as shown in Fig. 21(b)) decelerates the flow (also increases pressure) in the gap between the two rows. This reduction of momentum and increase of pressure prevent vortex shedding from the cylinders and thus make the flow almost steady.

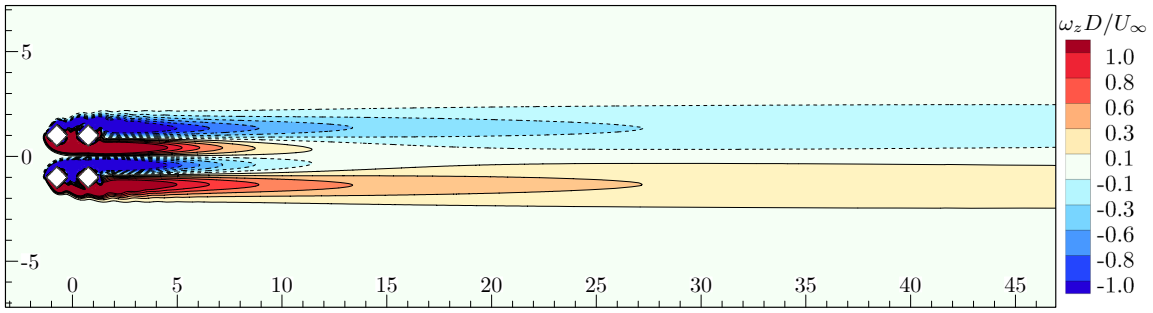


Figure 25: Vorticity contours of flow over multiple rotating square cylinders (Case 2).

5.7. Flow over 3D Rotating Square Cylinders

The first aim of this test is to demonstrate the method's capability for simulating 3D flows. Meanwhile, in the previous test we have noticed that a shift on the rotational direction of the cylinders has dramatically changed the 2D laminar flow fields. We are interested to see whether these changes persist in 3D at a much higher Reynolds number $Re_D = 5,000$, which serves as the second aim of this test. The mesh for this test is generated by extruding the previous 2D mesh in the spanwise (i.e., z) direction. The resulting domain has a size of πD in the spanwise direction, and is meshed into 32 intervals in this direction. All other parameters are exactly the same as those in the previous test. The simulation results are briefly summarized below.

We noticed suppression of vortex shedding and a much narrower wake in Case 2 of the previous test. This is somewhat still true for the 3D flows as shown in Figs. 26 and 27, where the flow fields are visualized using isosurfaces of Q-criterion [52] (denoted by $\mathcal{Q}$). From Fig. 26, we see vortex shedding from both the top and bottom sides of each row, and the interactions of these vortices lead to in a very wide and active wake. In contrast, in Fig.27, shear layers are seen on the two sides of each row and they do not break up until sufficiently far ($\approx 6D$) downstream of the cylinders, resulting in a much narrower and quieter wake. The present method is able to capture a lot of small turbulence structures, which is evidently benefited from the low dissipation nature of the method.

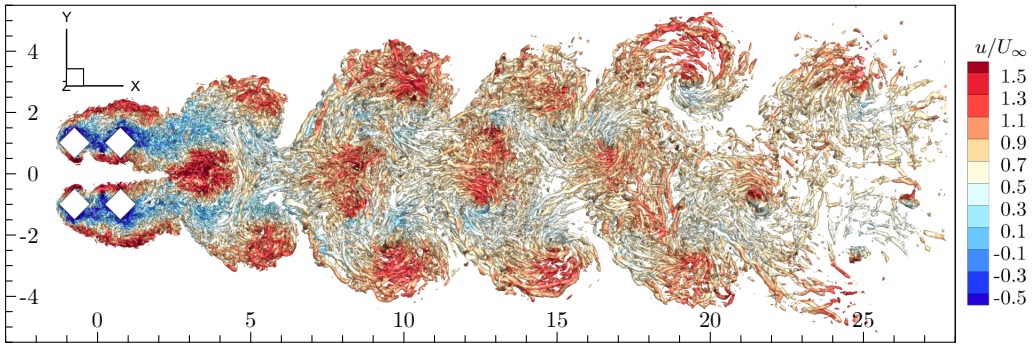


Figure 26: Isosurfaces of $\mathcal{Q}D^2/U_\infty^2 = 3$ for flow over multiple 3D rotating square cylinders (Case 1).

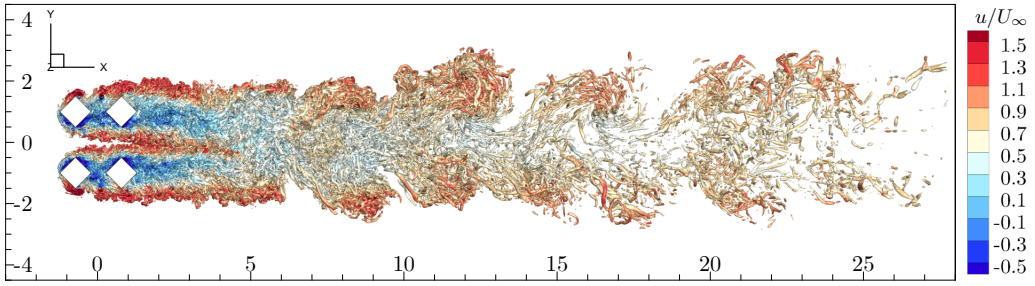


Figure 27: Isosurfaces of $\mathcal{Q}D^2/U_\infty^2 = 3$ for flow over multiple 3D rotating square cylinders (Case 2).

From the force coefficients in Figs. 28 and 29, it is seen that the magnitudes of the forces on the cylinders in the same column are no longer exactly the same. Nevertheless, they are still very close. On average, the two systems are still roughly lift-free. The drag coefficients reveal that Case 2 again has a significant drag reduction of about 50%, which is comparable to that in the 2D cases. To sum up, it is observed that no matter it is in 2D or 3D, by switching the signs of rotation from Case 1 to Case 2, we see a much quieter flow field, and a dramatic drag reduction of the system.

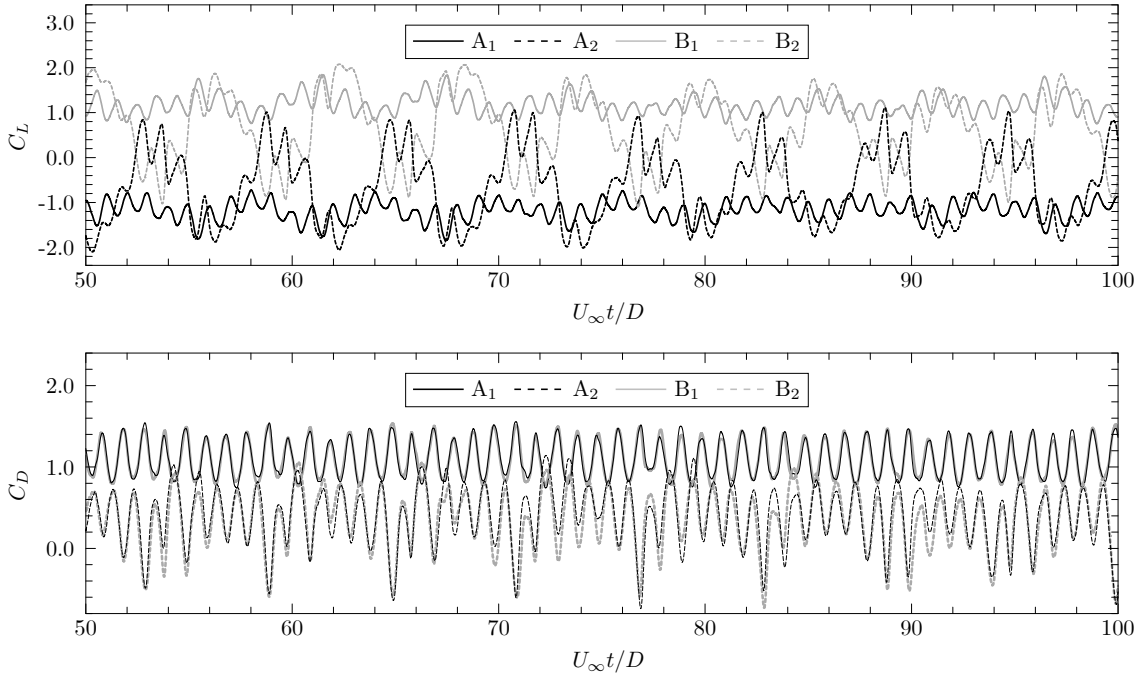


Figure 28: Lift and drag coefficients for flow over multiple 3D rotating square cylinders (Case 1).

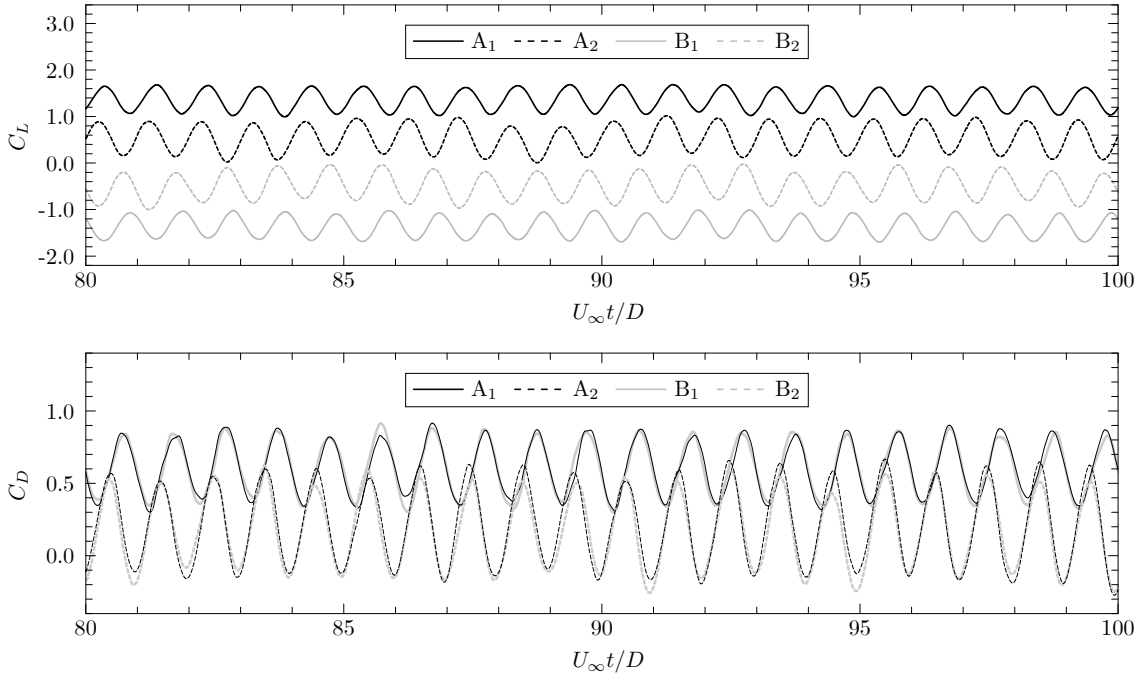


Figure 29: Lift and drag coefficients for flow over multiple 3D rotating square cylinders (Case 2).

6. Summary

The polynomial mortar and the transfinite mortar together provide a complete picture of the mortar method. These two mortar concepts are equivalent on linear meshes, but differ on curved meshes. More specifically, on curved meshes, the transfinite mortar introduces no geometric error, exactly satisfies the OS relations, has consistent spatial and temporal accuracies, is conservative, and guarantees outflow condition; the polynomial mortar, however, is shown to be less accurate and less conservative. On general nonuniform nonconforming curved meshes, both mortar types cannot directly satisfy free-stream preservation, which results in unavoidable numerical disturbances. We point out that this remains as an open problem for all polynomial-based high-order methods. Nevertheless, transfinite mortar is shown to generate much smaller numerical disturbances than polynomial mortar.

A sliding mesh method based on the transfinite mortar concept has been successfully developed, thoroughly tested and verified. It is shown that, in spite of the dynamic and curved nonconforming nature of the meshes, this method is arbitrarily high-order accurate in space, retains the temporal accuracy of a time-marching scheme, is conservative, and does not change flow characteristics. It sustains these properties even under very stringent, high-speed rotational/sliding conditions. Furthermore, it is shown that this method minimizes the rotational/sliding speed effects when it is equipped with a high-order temporal scheme. Although not strictly free-stream preserving, it approaches this property in an exponential way, and the mesh-induced numerical disturbances are shown to be negligibly small even on a very coarse mesh and moderate polynomial degrees. The implementation steps, the extension to 3D, and the treatment of singularity, are also discussed. The simulations of a matrix of rotating square cylinders in both 2D and 3D have demonstrated the method's capability of simultaneously handling multiple objects. This method is by far the most thoroughly studied and most accurate method for studying flows about rotating/sliding geometries. It can be further extended and applied to study a wide range of very challenging flow problems, such as propellers, wind turbines, stirred tanks, to name just a few. As a concluding remark, we emphasize that this method is readily implementable to many other high-order methods as well.

Acknowledgment

This work was supported by the Office of Naval Research (Grant N00014-20-1-2007) and the George Washington University.

Contribution statement

Bin Zhang: Conceptualization, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Chunlei Liang:** Funding acquisition.

Appendix A. Comparison of iso-parametric mapping and transfinite mapping

Transfinite mapping can minimizes geometric errors, whereas iso-parametric mapping always carries a truncation error but monolithically approaches transfinite mapping as the order increases. We compare these two types of mappings through a 1D and a 2D example. The 1D example is a circular arc with $\theta_1 = 0°$, $\theta_2 = 10°$, $R = 1$, and $\mathbf{x}_c = (0, 0)$. We calculate the radius difference ΔR between the exact radius and the numerical ones calculated using the coordinates from the mappings. Figure 30 reveals that the iso-parametric mapping represents the circular arc exactly only at the nodal points. For non-nodal points, the error overall decreases as the order increases. On the other hand, the transfinite mapping always represents the circular arc exactly with the errors always being at the level of machine precision.

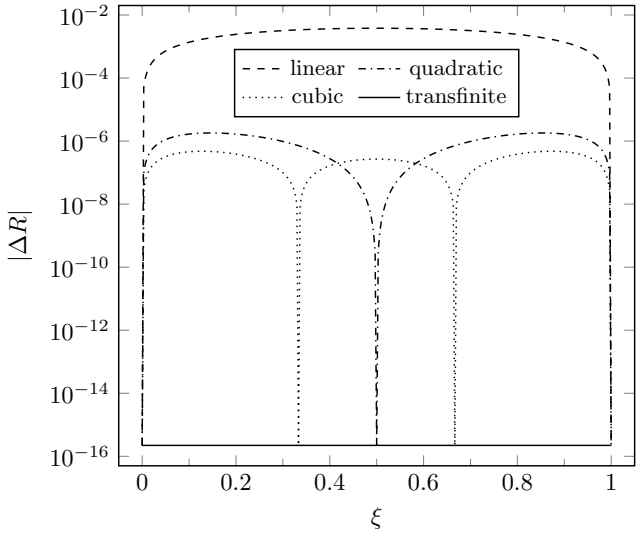


Figure 30: Comparison of iso-parametric mapping and transfinite mapping for approximating a circular arc.

The 2D geometry for this test is similar to that in Fig. 5, i.e., with three straight edges and one circular-arc edge. More specifically, we have chosen: $\mathbf{x}_1 = (R\cos\theta_1 + x_c, R\sin\theta_1 + y_c)$, $\mathbf{x}_2 = (R\cos\theta_2 + x_c, R\sin\theta_2 + y_c)$, $\mathbf{x}_3 = ((R/2)\tan\theta_2, R/2)$, $\mathbf{x}_4 = ((R/2)\tan\theta_1, R/2)$, where $R = 2$, $\theta_1 = -110°$, $\theta_2 = -70°$, and $(x_c, y_c) = (0, R)$. Denote the coordinates from linear, quadratic, cubic, and transfinite mappings as $\mathbf{x}_\mathrm{L}$, $\mathbf{x}_\mathrm{Q}$, $\mathbf{x}_\mathrm{C}$, and $\mathbf{x}_\mathrm{T}$, respectively. There is no direct way to measure which of these coordinates are more accurate. Instead, we take the transfinite mapping as a reference, and plot the contours of the L_2 norm of coordinate difference (i.e., $\|\mathbf{x}_* - \mathbf{x}_\mathrm{T}\|$, where '$*$' represents 'L', or 'Q', or 'C') in Fig. 31. The minimal differences are seen around the nodal points and along the straight edges. The reason is that these geometric components are represented in the same way in all the mappings including the transfinite mapping. Overall, the L_2 norm decreases as the order of the iso-parametric mapping increases, i.e., the iso-parametric mapping approaches the transfinite mapping as the order increases.

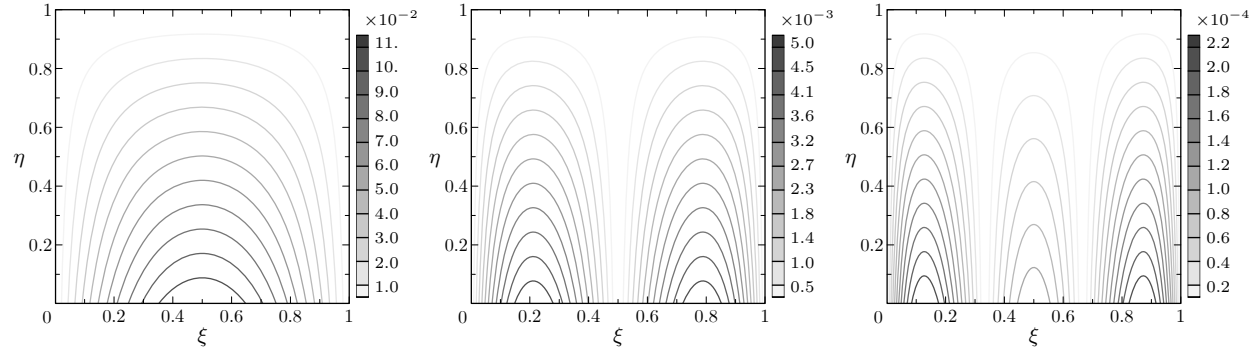


Figure 31: Difference between iso-parametric and transfinite mappings: left, $\|\mathbf{x}_\mathrm{L} - \mathbf{x}_\mathrm{T}\|$; middle, $\|\mathbf{x}_\mathrm{Q} - \mathbf{x}_\mathrm{T}\|$; right, $\|\mathbf{x}_\mathrm{C} - \mathbf{x}_\mathrm{T}\|$.

Appendix B. Proof of Global Conservation

Global conservation means that, without the presence of source in a domain, the net gain or loss of $\widetilde{\mathbf{Q}}$ over the whole domain is determined solely by fluxes through the boundaries. Inspired by the works in [15, 28], we prove in what follows that the present method is globally conservative.

Define the following quadrature weights w_i and w_j at the SPs such that

$$\int_0^1 \int_0^1 \phi(\xi,\eta)\mathrm{d}\xi\mathrm{d}\eta = \sum_{j=1}^{N}\sum_{i=1}^{N} \phi_{ij} w_i w_j, \quad \forall \phi \in \mathbf{P}_{N-1,N-1}. \tag{B.1}$$

Multiply the discretized equation (34) by the weights and sum over all the SPs,

$$\sum_{j=1}^{N}\sum_{i=1}^{N} \left.\frac{\partial \widetilde{\mathbf{Q}}}{\partial t}\right|_{ij} w_i w_j = -\sum_{j=1}^{N}\sum_{i=1}^{N} \left[\frac{\partial \widehat{\mathbf{F}}}{\partial \xi} + \frac{\partial \widehat{\mathbf{G}}}{\partial \eta}\right]_{ij} w_i w_j. \tag{B.2}$$

Since the quadrature is exact for $\mathbf{P}_{N-1,N-1}$ by its definition, and the terms $\partial\widetilde{\mathbf{Q}}/\partial t$, $\partial\widehat{\mathbf{F}}/\partial\xi$, $\partial\widehat{\mathbf{G}}/\partial\eta$ are in $\mathbf{P}_{N-1,N-1}$, (B.2) is thus equivalent to

$$\int_0^1 \int_0^1 \frac{\partial \widetilde{\mathbf{Q}}}{\partial t}\mathrm{d}\xi\mathrm{d}\eta = \int_0^1 \int_0^1 \left[\frac{\partial \widehat{\mathbf{F}}}{\partial \xi} + \frac{\partial \widehat{\mathbf{G}}}{\partial \eta}\right] \mathrm{d}\xi\mathrm{d}\eta. \tag{B.3}$$

Take the time derivative out, and apply the divergence theorem to the right hand side,

$$\frac{\partial}{\partial t}\int_0^1 \int_0^1 \widetilde{\mathbf{Q}}\mathrm{d}\xi\mathrm{d}\eta = \int_0^1 \widehat{\mathbf{F}}(0,\eta)\mathrm{d}\eta - \int_0^1 \widehat{\mathbf{F}}(1,\eta)\mathrm{d}\eta + \int_0^1 \widehat{\mathbf{G}}(\xi,0)\mathrm{d}\xi - \int_0^1 \widehat{\mathbf{G}}(\xi,1)\mathrm{d}\xi, \tag{B.4}$$

where, by definition, the corrected fluxes $\widehat{\mathbf{F}}$ and $\widehat{\mathbf{G}}$ take the common values on cell interfaces. For a conforming mesh, if we sum (B.4) over all the mesh elements, the interior flux terms cancel on cell interfaces, leaving only the flux terms on domain boundaries. This confirms that the change of $\widetilde{\mathbf{Q}}$ over the whole domain depends only on boundary fluxes, i.e., the scheme is globally conservative on a conforming mesh.

For nonconforming sliding mesh, we need to show that the total flux on the left side of a sliding interface equals that on the right side so that global conservation is satisfied. Since we have required (66) to hold for both inviscid and viscous fluxes with any h_j, and the h_j's form a basis of $\mathbf{P}_{N-1}$, the following relation thus holds,

$$\sum_{k=1}^{m} \int_{o_k}^{o_k+s_k} \left(\widetilde{F}^{\Omega}(\xi) - \widetilde{F}^{\Xi_k}(z)\right) \psi(\xi)\ \mathrm{d}\xi = 0, \quad \forall \psi \in \mathbf{P}_{N-1}, \tag{B.5}$$

Substitution of $\psi = 1 \in \mathbf{P}_{N-1}$ into the above equation gives

$$\sum_{k=1}^{m} \int_{o_k}^{o_k+s_k} \left(\widetilde{F}^{\Omega}(\xi) - \widetilde{F}^{\Xi_k}(z)\right) \mathrm{d}\xi = 0. \tag{B.6}$$

Rearrange the above equation to

$$\sum_{k=1}^{m} \int_{o_k}^{o_k+s_k} \widetilde{F}^{\Omega}(\xi)\mathrm{d}\xi = \sum_{k=1}^{m} \int_{o_k}^{o_k+s_k} \widetilde{F}^{\Xi_k}(z)\mathrm{d}\xi,$$

then change the variable of integration on the right hand side,

$$\int_0^1 \widetilde{F}^{\Omega}(\xi)\mathrm{d}\xi = \sum_{k=1}^{m} \int_0^1 s_k \widetilde{F}^{\Xi_k}(z)\mathrm{d}z,$$

and now consider the relation in (64) (which also applies to viscous flux),

$$\int_0^1 \widetilde{F}^{\Omega}(\xi)\mathrm{d}\xi = \sum_{k=1}^{m} \int_0^1 \breve{F}^{\Xi_k}(z)\mathrm{d}z. \tag{B.7}$$

The relation in (B.7) is valid for any sliding cell face Ω, and it says that the flux on a sliding cell face equals the total flux on all its mortars. Since each mortar is unique and does not overlap with other mortars (see Fig. 3), summing (B.7) over all the cell faces on the left side of a sliding interface will lead to the conclusion that the total flux on the left side of a sliding interface equals the total flux on all the mortars. The same conclusion also holds on the right side of a sliding interface. Therefore, the total flux on the left side of a sliding interface equals that on the right side, i.e., the scheme satisfies global conservation. It is worth noting that, in Eq. (B.7), the variable $\breve{F}^{\Xi_k}$ (see Eq. (63) for the definition) has already "absorbed" the exact metrics (such as the exact OS relations in Eqs. (43)-(45), and the exact normal in Eq. (61)) of the transfinite mortar. The Introduction of this new variable/notation has greatly simplified the proof. We monitored the net fluxes on each nonconforming interface for all the tests, and it was observed that the values are always at the level of machine precision.

Appendix C. Proof of Outflow Condition

In a hyperbolic system, the propagation of waves should not be affected by downwind signals, which is called the outflow condition. When it comes to our case, satisfying the outflow condition means that if we project a variable from a cell face to its mortars, and then immediately project back, the value of the variable should not change. In what follows, we prove that the present sliding-mesh method satisfies this condition. Since the two projection methods in Sec. 4.3.3 are equivalent, we only show the poof on the first method for simplicity.

Let ϕ^{Ω} denote a variable on a cell face, ϕ^{Ξ_k} the projected variable on the k-th mortar, and $\phi^{*\Omega}$ the variable from back projection. Then, the problem becomes proving that $\phi^{*\Omega} = \phi^{\Omega}$.

Start with the following polynomial representations of these variables,

$$\phi^{\Omega}(\xi) = \sum_{i=1}^{N} \phi_i^{\Omega} h_i(\xi), \quad \phi^{\Xi_k}(z) = \sum_{i=1}^{N} \phi_i^{\Xi_k} h_i(z), \quad \phi^{*\Omega}(\xi) = \sum_{i=1}^{N} \phi_i^{*\Omega} h_i(\xi). \tag{C.1}$$

The projection from cell face to mortar follows (48),

$$\int_0^1 \left(\phi^{\Xi_k}(z) - \phi^{\Omega}(\xi)\right) h_j(z)\ \mathrm{d}z = 0, \quad \forall j = 1, 2, ..., N, \tag{C.2}$$

from which we can get the projected variable on the mortar,

$$\boldsymbol{\phi}^{\Xi_k} = \mathbf{P}^{\Omega \to \Xi_k} \boldsymbol{\phi}^{\Omega} = \mathbf{M}^{-1} \mathbf{S}^{\Omega \to \Xi_k} \boldsymbol{\phi}^{\Omega}. \tag{C.3}$$

Now, project ϕ^{Ξ_k} back to the cell face following (56),

$$\sum_{k=1}^{m} \int_{o_k}^{o_k + s_k} \left(\phi^{*\Omega}(\xi) - \phi^{\Xi_k}(z)\right) h_j(\xi)\ \mathrm{d}\xi = 0, \quad \forall j = 1, 2, ..., N, \tag{C.4}$$

from which we can get the variable from back projection,

$$\boldsymbol{\phi}^{*\Omega} = \sum_{k=1}^{m} \mathbf{P}^{\Xi_k \to \Omega} \boldsymbol{\phi}^{\Xi_k} = \sum_{k=1}^{m} s_k \mathbf{M}^{-1} \mathbf{S}^{\Xi_k \to \Omega} \boldsymbol{\phi}^{\Xi_k}. \tag{C.5}$$

Since the the h_j's form a basis for $\mathbf{P}_{N-1}$, from (C.2) we can get

$$\int_0^1 \left(\phi^{\Xi_k}(z) - \phi^{\Omega}(\xi)\right) \psi\ \mathrm{d}z = 0, \quad \forall \psi \in \mathbf{P}_{N-1}. \tag{C.6}$$

Because $h_j(\xi) = h_j(o_k + s_k z) \in \mathbf{P}_{N-1}$, setting $\psi = h_j(\xi)$ gives

$$\int_0^1 \left(\phi^{\Xi_k}(z) - \phi^{\Omega}(\xi)\right) h_j(\xi)\ \mathrm{d}z = 0,$$

now rearrange the terms and change the variable of integration,

$$\int_{o_k}^{o_k+s_k} \phi^{\Omega}(\xi) h_j(\xi)\ \mathrm{d}\xi = s_k \int_0^1 \phi^{\Xi_k}(z) h_j(\xi)\ \mathrm{d}z. \tag{C.7}$$

Apply (C.7) to all the mortars of Ω and sum them up,

$$\sum_{k=1}^{m} \int_{o_k}^{o_k+s_k} \phi^{\Omega}(\xi) h_j(\xi)\ \mathrm{d}\xi = \sum_{k=1}^{m} s_k \int_0^1 \phi^{\Xi_k}(z) h_j(\xi)\ \mathrm{d}z,$$

$$\int_0^1 \phi^{\Omega}(\xi) h_j(\xi)\ \mathrm{d}\xi = \sum_{k=1}^{m} s_k \int_0^1 \phi^{\Xi_k}(z) h_j(\xi)\ \mathrm{d}z, \tag{C.8}$$

At the same time, rearranging (C.4) and changing the variable of integration will give

$$\int_0^1 \phi^{*\Omega}(\xi) h_j(\xi) \mathrm{d}\xi = \sum_{k=1}^{m} s_k \int_0^1 \phi^{\Xi_k}(z) h_j(\xi)\ \mathrm{d}z. \tag{C.9}$$

Note that (C.8) and (C.9) have exactly the same right hand side, therefore

$$\int_0^1 \phi^{*\Omega}(\xi) h_j(\xi) \mathrm{d}\xi = \int_0^1 \phi^{\Omega}(\xi) h_j(\xi) \mathrm{d}\xi, \quad \forall j = 1, 2, ..., N, \tag{C.10}$$

which is equivalent to the following equation system,

$$\mathbf{M}\boldsymbol{\phi}^{*\Omega} = \mathbf{M}\boldsymbol{\phi}^{\Omega}.$$

Because $\mathbf{M}$ is invertible, we therefore have

$$\boldsymbol{\phi}^{*\Omega} = \mathbf{M}^{-1}\mathbf{M}\boldsymbol{\phi}^{\Omega},$$

that is

$$\boldsymbol{\phi}^{*\Omega} = \boldsymbol{\phi}^{\Omega}, \tag{C.11}$$

which is exactly what we were expecting, and the present method therefore satisfies the outflow condition. If we substitute (C.3) into (C.5) and take (C.11) into account, we have the following relation

$$\boldsymbol{\phi}^{*\Omega} = \sum_{k=1}^{m} \mathbf{P}^{\Xi_k \to \Omega} \mathbf{P}^{\Omega \to \Xi_k} \boldsymbol{\phi}^{\Omega} = \boldsymbol{\phi}^{\Omega}, \tag{C.12}$$

which readily reveals a property of the projection matrices,

$$\sum_{k=1}^{m} \mathbf{P}^{\Xi_k \to \Omega} \mathbf{P}^{\Omega \to \Xi_k} = \mathbf{I}, \tag{C.13}$$

where $\mathbf{I}$ is the identity matrix.

We have numerically tested (C.13) on many cases, and it holds perfectly without exception. As it was noted in [28] that satisfaction of the outflow condition ensures that a method does not change the characteristics of the flow field, and therefore there should not be a downgrade on the maximum allowable time step size compared with the original method on conforming mesh. This is consistent with our observation through many tests that the present method does not affect the stability of the original conforming-mesh method.

References

[1] A. Bakker, R. D. LaRoche, M.-H. Wang, R. V. Calabrese, Sliding mesh simulation of laminar flow in stirred reactors, Chemical Engineering Research and Design 75 (1) (1997) 42–44.

[2] R. Steijl, G. Barakos, Sliding mesh algorithm for CFD analysis of helicopter rotor-fuselage aerodynamics, International Journal for Numerical Methods in Fluids 58 (5) (2008) 527–549.

[3] T. Kinsey, G. Dumas, Parametric study of an oscillating airfoil in a power-extraction regime, AIAA Journal 46 (2008) 1318–1330.

[4] M. W. Sarwar, T. Ishihara, Numerical study on suppression of vortex-induced vibrations of box girder bridge section by aerodynamic countermeasures, Journal of Wind Engineering and Industrial Aerodynamics 98 (12) (2010) 701–711.

[5] J. L. Steger, F. C. Dougherty, J. A. Benek, A Chimera grid scheme, in: Proceedings of the Applied Mechanics, Bioengineering, and Fluids Engineering Conference, Houston, TX, 59–69, 1983.

[6] R. Mittal, G. Iaccarino, Immersed boundary methods, Annual Review of Fluid Mechanics 37 (2005) 239–261.

[7] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, et al., High-order CFD methods: current status and perspective, International Journal for Numerical Methods in Fluids 72 (8) (2013) 811–845.

[8] W. H. Reed, T. R. Hill, Triangular mesh methods for the neutron transport equation, Tech. Rep. LA-UR-73-479; CONF-730414-2, Los Alamos Scientific Laboratory, 1973.

[9] B. Cockburn, G. E. Karniadakis, C.-W. Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, vol. 11 of *Lecture Notes in Computational Science and Engineering*, Springer, New York, 2011.

[10] A. T. Patera, A spectral element method for fluid dynamics: laminar flow in a channel expansion, Journal of Computational Physics 54 (3) (1984) 468–488.

[11] G. E. Karniadakis, S. J. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, Oxford University Press, Oxford, 2 edn., 2005.

[12] D. A. Kopriva, Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers, Springer Science & Business Media, 2009.

[13] Z. J. Wang, Spectral (finite) volume method for conservation laws on unstructured grids: basic formulation, Journal of Computational Physics 178 (2002) 210–251.

[14] Z. J. Wang, Y. Liu, Spectral (finite) volume method for conservation laws on unstructured grids: II. Extension to two-dimensional scalar equation, Journal of Computational Physics 179 (2) (2002) 665–697.

[15] D. A. Kopriva, J. H. Kolias, A conservative staggered-grid Chebyshev multidomain method for compressible flows, Journal of Computational Physics 125 (1996) 244–261.

[16] D. A. Kopriva, A staggered-grid multidomain spectral method for the compressible Navier-Stokes equations, Journal of Computational Physics 143 (1998) 125–158.

[17] Y. Liu, M. Vinokur, Z. J. Wang, Spectral difference method for unstructured grids I: Basic formulation, Journal of Computational Physics 216 (2006) 780–801.

[18] Z. J. Wang, Y. Liu, G. May, A. Jameson, Spectral difference method for unstructured grids II: Extension to the Euler equations, Journal of Scientific Computing 32 (2007) 45–71.

[19] A. Balan, G. May, J. Schöberl, A stable high-order spectral difference method for hyperbolic conservation laws on triangular elements, Journal of Computational Physics 231 (5) (2012) 2359–2375.

[20] H. T. Huynh, A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods, AIAA paper 2007-4079, 2007.

[21] H. T. Huynh, A reconstruction approach to high-order schemes including discontinuous Galerkin for diffusion, AIAA paper 2009-403, 2009.

[22] Z. J. Wang, H. Gao, A unifying lifting collocation penalty formulation including the discontinuous Galerkin, spectral volume/difference methods for conservation laws on mixed grids, Journal of Computational Physics 228 (2009) 8161–8186.

[23] A. Jameson, P. E. Vincent, P. Castonguay, On the non-linear stability of flux reconstruction schemes, Journal of Scientific Computing 50 (2) (2012) 434–445.

[24] Z. J. Wang, H. T. Huynh, A review of flux reconstruction or correction procedure via reconstruction method for the Navier-Stokes equations, Mechanical Engineering Reviews 3 (1) (2016) 15–00475.

[25] E. Ferrer, R. Willden, A high order discontinuous Galerkin-Fourier incompressible 3D Navier-Stokes solver with rotating sliding meshes, Journal of Computational Physics 231 (21) (2012) 7037–7056.

[26] L. Ramírez, C. Foulquié, X. Nogueira, S. Khelladi, J.-C. Chassaing, I. Colominas, New high-resolution-preserving sliding mesh techniques for higher-order finite volume schemes, Computers & Fluids 118 (2015) 114–130.

[27] C. A. Mavriplis, Nonconforming Discretizations and a Posteriori Error Estinates for Adaptive Spectral Element Techniques, Ph.D. thesis, M.I.T., 1989.

[28] D. A. Kopriva, A conservative staggered-grid Chebyshev multidomain method for compressible flows. II. A semi-structured method, Journal of Computational Physics 128 (1996) 475–488.

[29] D. A. Kopriva, S. L. Woodruff, M. Y. Hussaini, Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method, International journal for numerical methods in engineering 53 (1) (2002) 105–122.

[30] B. Zhang, C. Liang, A simple, efficient, and high-order accurate curved sliding-mesh interface approach to spectral difference method on coupled rotating and stationary domains, Journal of Computational Physics 295 (2015) 147–160.

[31] B. Zhang, C. Liang, J. Yang, Y. Rong, A 2D parallel high-order sliding and deforming spectral difference method, Computers & Fluids 139 (2016) 184–196.

[32] B. Zhang, C. Liang, A high-order sliding-mesh spectral difference solver for simulating unsteady flows around rotating objects, in: 31st Symposium on Naval Hydrodynamics, Monterey, CA, 2016.

[33] B. Zhang, Z. Qiu, C. Liang, A flux reconstruction method with nonuniform sliding-mesh interfaces for simulating rotating flows, AIAA paper 2018-1094, 2018.

[34] Z. Qiu, B. Zhang, C. Liang, M. Xu, A high-order solver for simulating vortex-induced vibrations using the sliding-mesh spectral difference method and hybrid grids, International Journal for Numerical Methods in Fluids 90 (4) (2019) 171–194.

[35] B. Zhang, C. Liang, High-order numerical simulation of flows over rotating cylinders of various cross-sectional shapes, AIAA paper 2019-3430, 2019.

[36] B. Zhang, C. Liang, High-order numerical simulation of flapping wing for energy harvesting, AIAA paper 2019-3338, 2019.

[37] B. Zhang, C. Ding, C. Liang, High-order implicit large-eddy simulation of flow over a marine propeller, Computers & Fluids 224 (2021) 104967.

[38] J. Yang, B. Zhang, C. Liang, Y. Rong, A high-order flux reconstruction method with adaptive mesh refinement and artificial diffusivity on unstructured moving/deforming mesh for shock capturing, Computers & Fluids 139 (2016) 17–35.

[39] I. Ergatoudis, B. M. Irons, O. C. Zienkiewicz, Curved, isoparametric, "quadrilateral" elements for finite element analysis, International Journal of Solids and Structures 4 (1) (1968) 31–42.

[40] V. V. Rusanov, Calculation of interaction of non-steady shock waves with obstacles, Journal of Computational and Mathematical Physics USSR 1 (1961) 267–279.

[41] J. F. B. M. Kraaijevanger, Contractivity of Runge-Kutta methods, BIT Numerical Mathematics 31 (3) (1991) 482–528.

[42] S. Ruuth, Global optimization of explicit strong-stability-preserving Runge-Kutta methods, Mathematics of Computation 75 (253) (2006) 183–207.

[43] S. Gottlieb, S. J. Ruuth, Optimal strong-stability-preserving time-stepping schemes with fast downwind spatial discretizations, Journal of Scientific Computing 27 (1-3) (2006) 289–303.

[44] P. D. Thomas, C. K. Lombard, Geometric conservation law and its application to flow computations on moving grids, AIAA Journal 17 (1979) 1030–1037.

[45] C. A. A. Minoli, D. A. Kopriva, Discontinuous Galerkin spectral element approximations on moving meshes, Journal of Computational Physics 230 (5) (2011) 1876–1902.

[46] W. J. Gordon, C. A. Hall, Transfinite element methods: blending-function interpolation over arbitrary curved element domains, Numerische Mathematik 21 (2) (1973) 109–129.

[47] W. J. Gordon, C. A. Hall, Construction of curvilinear co-ordinate systems and applications to mesh generation, International Journal for Numerical Methods in Engineering 7 (4) (1973) 461–477.

[48] R. J. Spiteri, S. J. Ruuth, A new class of optimal high-order strong-stability-preserving time discretization methods, SIAM Journal on Numerical Analysis 40 (2002) 469–491.

[49] D. I. Ketcheson, Highly efficient strong stability-preserving Runge–Kutta methods with low-storage implementations, SIAM Journal on Scientific Computing 30 (4) (2008) 2113–2136.

[50] D. A. Kopriva, Metric identities and the discontinuous spectral element method on curvilinear meshes, Journal of Scientific Computing 26 (3) (2006) 301.

[51] D. A. Kopriva, F. J. Hindenlang, T. Bolemann, G. J. Gassner, Free-stream preservation for curved geometrically non-conforming discontinuous Galerkin spectral elements, Journal of Scientific Computing 79 (3) (2019) 1389–1408.

[52] J. Hunt, A. Wray, P. Moin, Eddies, streams, and convergence zones in turbulent flows, in: Proceedings of CTR Summer Program, Center for Turbulence Research, 193–208, 1988.