Journal of Computational Physics 295 (2015) 147–160

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A simple, efficient, and high-order accurate curved sliding-mesh interface approach to spectral difference method on coupled rotating and stationary domains

Bin Zhang, Chunlei Liang [*]

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

A R T I C L E   I N F O

Article history:
Received 27 September 2014
Received in revised form 6 April 2015
Accepted 8 April 2015
Available online 16 April 2015

Keywords:
Spectral difference method
Rotating mesh
Sliding mesh
High-order methods
Unstructured grid

A B S T R A C T

This paper presents a simple, efficient, and high-order accurate sliding-mesh interface approach to the spectral difference (SD) method. We demonstrate the approach by solving the two-dimensional compressible Navier–Stokes equations on quadrilateral grids. This approach is an extension of the straight mortar method originally designed for stationary domains [7,8]. Our sliding method creates curved dynamic mortars on sliding-mesh interfaces to couple rotating and stationary domains. On the nonconforming sliding-mesh interfaces, the related variables are first projected from cell faces to mortars to compute common fluxes, and then the common fluxes are projected back from the mortars to the cell faces to ensure conservation. To verify the spatial order of accuracy of the sliding-mesh spectral difference (SSD) method, both inviscid and viscous flow cases are tested. It is shown that the SSD method preserves the high-order accuracy of the SD method. Meanwhile, the SSD method is found to be very efficient in terms of computational cost. This novel sliding-mesh interface method is very suitable for parallel processing with domain decomposition. It can be applied to a wide range of problems, such as the hydrodynamics of marine propellers, the aerodynamics of rotorcraft, wind turbines, and oscillating wing power generators, etc.

1. Introduction

High-order (third and above) numerical methods are becoming more and more popular in recent years due to their capability of producing more accurate solutions on relatively coarse grid [31]. The spectral difference (SD) method is one discontinuous high-order method for solving the conservation laws on unstructured grids [15,32,28,11]. This method is an extension of the staggered multi-domain high-order method originally designed by Kopriva and Kolias [9]. It was shown that the SD method also has strong connection with the Flux Reconstruction/Correction Procedure via Reconstruction (FR/CPR) methods [5], and it shares similarity with the quadrature-free discontinuous Galerkin method [19]. The stability of a particular choice of flux points for the SD method was proved by Jameson [6] for the one-dimensional linear wave equation. Although the proof has not been generalized to higher-dimensional tensor-product elements, we have not observed numerical instability from several successful turbulent flow simulations [14,1,22].

[*] Corresponding author. Tel.: +1 202 994 7073.
E-mail addresses: bzh@gwu.edu (B. Zhang), chliang@gwu.edu (C. Liang).

http://dx.doi.org/10.1016/j.jcp.2015.04.006
0021-9991/© 2015 Elsevier Inc. All rights reserved.

We have seen more and more applications of the SD method to realistic flow simulations, for example, for large eddy simulations on fixed grids [14,1,22,25,24,16]. The SD method is also particularly promising for simulating vortex-dominated flows on moving and deforming grids [23,34]. Liang et al. [12] extended the SD method for simulating two-dimensional unsteady flows around a plunging or pitching airfoil. DeJong and Liang [2] studied three-dimensional vortex induced vibrations using the SD method.

However, when the mesh undergoes very large rotation motion, such as for flows around rotating propellers or passing a flapping wing with very large pitching angles, remeshing [29,30] is required. Our goal is to involve the minimum number of remeshing and simultaneously preserve the high-order accuracy of the SD method. This motivates us to develop a new approach to the SD method for coupled rotating and stationary domains with sliding-mesh interfaces. In our approach, both inviscid and viscous fluxes on the sliding-mesh interfaces are constructed using a newly developed curved dynamic mortar method. The mortar method on fixed grids was originally proposed for incompressible flows by Mavriplis [18]. Kopriva [7,8] proved the conservation property of the mortar method for the compressible flow equations and applied it to the compressible Euler and Navier–Stokes equations on stationary domains using structured grids. In this paper, we show that our sliding-mesh approach is as simple as those designed for low-order numerical methods [33,20] while preserving the high-order accuracy of the SD method. This simple but novel sliding-mesh spectral difference (SSD) method can have a wide range of applications, such as marine propulsor hydrodynamics, rotorcraft aerodynamics, wind turbine wake dynamics, and oscillating wing power generators.

The paper is organized as follows: Section 2 gives the two-dimensional compressible Navier–Stokes equations on stationary and rotating domains. Section 3 reviews the SD method and presents the SSD method in detail. Verification studies and applications are reported in Section 4. Section 5 concludes the paper.

2. The governing equations

2.1. The compressible Navier–Stokes equations on stationary domain

We consider the two-dimensional unsteady compressible Navier–Stokes equations in conservative form,

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

where $\mathbf{Q}$ is the vector of conservative variables, $\mathbf{F}$ and $\mathbf{G}$ are the x and the y flux vectors. These terms have the following expressions,

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

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

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

where ρ is the fluid density, u and v are the x and the y velocities, E is the total energy per volume defined as $E=p/(\gamma-1)+\frac{1}{2}\rho(u^2+v^2)$, p is the pressure, γ is the ratio of specific heats and is set to 1.4 (i.e., the typical value for the air in standard conditions).

As shown in Eqs. (3) and (4), the fluxes have been divided into inviscid and viscous parts. The inviscid fluxes are only functions of conservative variables, which are

$$\mathbf{F}_{inv}=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ (E+p)u\end{bmatrix},\quad \mathbf{G}_{inv}=\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ (E+p)v\end{bmatrix}. \tag{5}$$

The viscous fluxes are functions of the conservative variables as well as their gradients. They have the following expressions,

$$\mathbf{F}_{vis}=-\begin{bmatrix}0\\ \tau_{xx}\\ \tau_{yx}\\ u\tau_{xx}+v\tau_{yx}+\mathrm{k}T_x\end{bmatrix},\quad \mathbf{G}_{vis}=-\begin{bmatrix}0\\ \tau_{xy}\\ \tau_{yy}\\ u\tau_{xy}+v\tau_{yy}+\mathrm{k}T_y\end{bmatrix}, \tag{6}$$

where τ_{ij} is the shear stress tensor and is related to the velocity gradients as $\tau_{ij}=\mu(u_{i,j}+u_{j,i})+\lambda\delta_{ij}u_{k,k}$, μ is the dynamic viscosity, $\lambda=-2/3\mu$ based on Stokes' hypothesis, δ_{ij} is the Kronecker delta, k is the thermal conductivity, T is the temperature which is related to density and pressure through the ideal gas law $p=\rho RT$, where R is the gas constant.

2.2. The compressible Navier–Stokes equations on rotating domain

On the rotating domains, we implement a simplified equation which is equivalent to the Arbitrary Lagrange–Eulerian (ALE) [4] form of Eq. (1). Due to grid motion, the inviscid fluxes are modified to take the following forms,

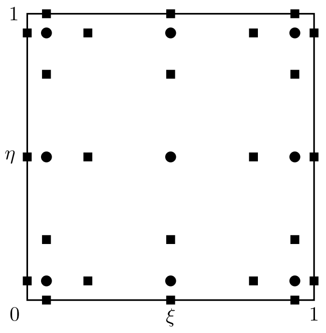

Fig. 1. Schematic of the distribution of solution points (circles) and flux points (squares) for a third-order SD scheme.

$$\mathbf{F}_{inv} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ (E+p)u \end{bmatrix} - u_g \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \quad \mathbf{G}_{inv} = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ (E+p)v \end{bmatrix} - v_g \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \tag{7}$$

where u_g and v_g are the x and the y grid velocities, respectively. The viscous fluxes and all other variables are uninfluenced and take the same expressions as those in the previous section.

For a domain rotating at angular velocity $\boldsymbol{\omega}$, the grid velocities are $(u_g, v_g) = \boldsymbol{\omega} \times \mathbf{r}$, where $\mathbf{r}$ is the position vector with respect to rotating center. For all test cases in the present study, $\boldsymbol{\omega}$ is known as a priori, thus grid velocities and coordinates are updated analytically on the rotating domains.

2.3. The transformed equations

As will be discussed in the next section, we map each quadrilateral grid cell in the physical domain to a standard square element in a computational domain. This mapping facilitates the construction of solution and flux polynomials. As a result, we only need to solve a set of transformed equations within each standard element. Let us assume that the physical coordinates (x, y) are mapped to the computational ones (ξ, η) through a transformation: $x = x(\xi, \eta)$, $y = y(\xi, \eta)$. It can be shown that Eq. (1) will take the following conservative form after coordinates transformation,

$$\frac{\partial \widetilde{\mathbf{Q}}}{\partial t} + \frac{\partial \widetilde{\mathbf{F}}}{\partial \xi} + \frac{\partial \widetilde{\mathbf{G}}}{\partial \eta} = 0, \tag{8}$$

where $\widetilde{\mathbf{Q}} = |\mathcal{J}|\mathbf{Q}$, and the transformed fluxes $\widetilde{\mathbf{F}}$, $\widetilde{\mathbf{G}}$ are related to the physical ones as

$$\begin{pmatrix} \widetilde{\mathbf{F}} \\ \widetilde{\mathbf{G}} \end{pmatrix} = |\mathcal{J}|\mathcal{J}^{-1} \begin{pmatrix} \mathbf{F} \\ \mathbf{G} \end{pmatrix}, \tag{9}$$

where $|\mathcal{J}|$ is determinant of the Jacobian matrix, and $\mathcal{J}^{-1}$ is the inverse Jacobian matrix,

$$|\mathcal{J}| = \left| \frac{\partial(x, y)}{\partial(\xi, \eta)} \right| = \begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix} = x_\xi y_\eta - x_\eta y_\xi, \tag{10}$$

$$\mathcal{J}^{-1} = \frac{\partial(\xi, \eta)}{\partial(x, y)} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \frac{1}{|\mathcal{J}|} \begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix}. \tag{11}$$

3. Numerical methods

In this section, we first give a brief review of the SD method. Subsequently, we describe a newly formulated sliding-mesh interface technique that is built on the SD formulation. For temporal discretization, an explicit strong stability preserving Runge–Kutta method [27] is used for all computations throughout this paper.

3.1. The SD method

For SD method on quadrilateral grids, we first transform each cell in the physical domain to a standard square element $(0 \le \xi \le 1, 0 \le \eta \le 1)$ in the computational domain. The transformation can be done through iso-parametric mapping. As was reported by Liang et al. [10,13], using linear cell defined by four nodes is not sufficient for problems involving curved boundaries. High-order cubic cell with twelve nodes are used along the curved boundaries to ensure stability and accuracy in the present study.

After the mapping, solution points (SPs) and flux points (FPs) are defined on each standard element as shown in Fig. 1 for a third-order scheme. For an N-th order SD method, N SPs are required along each coordinate direction to construct

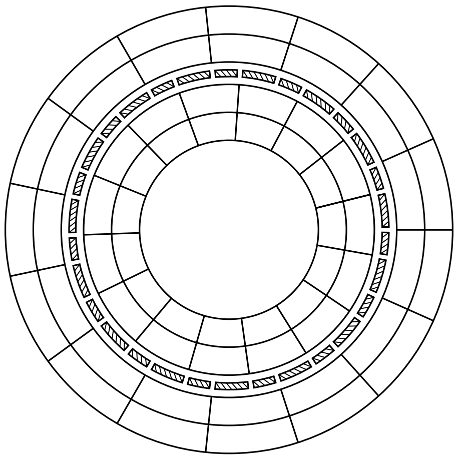

Fig. 2. Schematic of the distribution of mortars (hatched) between a rotating and a stationary meshes.

degree $(N-1)$ solution polynomials, and $(N+1)$ FPs are employed in each direction to construct degree N flux polynomials. In the current implementation, the SPs: X_s, where $s = 1, 2, \dots, N$, are chosen as N Chebyshev–Gauss points. The FPs: $X_{s+1/2}$, where $s = 0, 1, 2, \dots, N$, are chosen as $(N-1)$ Legendre–Gauss points plus two end points to align in a staggered fashion with the SPs.

To construct solution and flux polynomials, the following Lagrange bases at the SPs and FPs are used,

$$h_i(X) = \prod_{s=1, s\neq i}^{N} \left(\frac{X - X_s}{X_i - X_s} \right), \tag{12}$$

$$l_{i+1/2}(X) = \prod_{s=0, s\neq i}^{N} \left(\frac{X - X_{s+1/2}}{X_{i+1/2} - X_{s+1/2}} \right). \tag{13}$$

The solution and the fluxes within each element are simply tensor products of the Lagrange bases,

$$\mathbf{Q}(\xi, \eta) = \sum_{j=1}^{N} \sum_{i=1}^{N} \frac{\widetilde{\mathbf{Q}}_{i,j}}{|\mathcal{J}_{i,j}|} h_i(\xi) \cdot h_j(\eta), \tag{14}$$

$$\widetilde{\mathbf{F}}(\xi, \eta) = \sum_{j=0}^{N} \sum_{i=0}^{N} \widetilde{\mathbf{F}}_{i+1/2,j} l_{i+1/2}(\xi) \cdot h_j(\eta), \tag{15}$$

$$\widetilde{\mathbf{G}}(\xi, \eta) = \sum_{j=0}^{N} \sum_{i=0}^{N} \widetilde{\mathbf{G}}_{i,j+1/2} h_i(\xi) \cdot l_{j+1/2}(\eta). \tag{16}$$

The above reconstructed solution and fluxes are only element-wise continuous, but discontinuous across cell interfaces. A Riemann solver is employed to compute the common inviscid fluxes at cell interfaces to ensure conservation. In the current implementation, the Rusanov solver [26] has been used for this purpose. The common viscous fluxes are computed from common solution and common gradients, and the detailed steps can be found in previous papers by Liang et al. [10,13].

3.2. The sliding-mesh interface approach

Sliding-mesh interfaces are formed between rotating and stationary meshes. The simplest situation involves only one rotating mesh and one stationary mesh as shown in Fig. 2. The inner mesh can rotate while the outer is fixed, or vice versa. Communication between the stationary and the rotating meshes is realized through “mortars”. To make the explanation intuitive, we have scaled the inner mesh in order to place mortars in between the two coupled meshes.

The mortars are arranged in a counterclockwise order. We refer the inner mesh as left (L) and the outer mesh as right (R) with respect to the mortars. To facilitate code implementation and reduce computational cost, cell faces on both sides of the sliding-mesh interface have been uniformly meshed. A closer look at Fig. 2 reveals how mortars and cell faces on the sliding-mesh interface are connected: at each time instant, a cell face is connected to two mortars, and each mortar is associated with one left and one right cell faces. This cell face and mortar connectivity needs to be updated at every stage of the Runge–Kutta time-stepping method. As was discussed in [7] for stationary grid, each cell face can have more than two mortars. Thus, our sliding-mesh interface method can also be extended to non-uniform meshes.

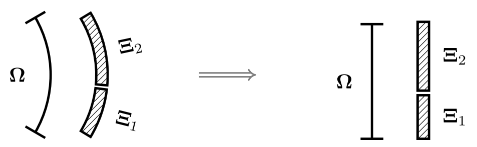

Fig. 3. Mapping of curved cell face and mortars to straight ones: left, curved face and mortars in physical domain; right, straight face and mortars in computational domain.

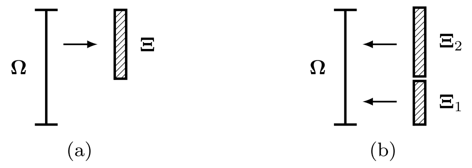

Fig. 4. Projection between face and mortar: (a) from left face to left side of mortar, (b) from two mortars back to the associated left face.

Fig. 3 shows a cell face Ω and the attached two mortars Ξ_1 and Ξ_2. Each curved mortar is mapped to a straight edge $0 \le z \le 1$ through $1D$ iso-parametric mapping. Face Ω is mapped to a straight edge $0 \le \xi \le 1$ when the associated cell is mapped to a standard square element, thus no extra mapping is required. ξ and z are related by

$$\xi = o(t) + s(t)z, \tag{17}$$

where $o(t)$ is the offset of the mortar relative to the bottom node of Ω at time t, and $s(t)$ is the relative scaling. For the example shown in Fig. 3, we have $o_1 = 0$ and $s_1 = L^{\Xi_1}/L^{\Omega}$ for Ξ_1, $o_2 = L^{\Xi_1}/L^{\Omega}$ and $s_2 = L^{\Xi_2}/L^{\Omega}$ for Ξ_2, where L denotes the physical length of the face or the mortar elements.

According to Eq. (14), the solutions on Ω can be represented as

$$\mathbf{Q}^{\Omega} = \sum_{i=1}^{N} \mathbf{Q}_i^{\Omega} h_i(\xi), \tag{18}$$

where $\mathbf{Q}_i^{\Omega}$ represents solution at the i-th SP on Ω, and h_i is the Lagrange basis defined in Eq. (12). If we define the same set of SPs on $0 \le z \le 1$ for each mortar, then the solutions on each mortar element can be reconstructed similarly as

$$\mathbf{Q}^{\Xi} = \sum_{i=1}^{N} \mathbf{Q}_i^{\Xi} h_i(z), \tag{19}$$

where $\mathbf{Q}_i^{\Xi}$ is the solution at the i-th SP on a mortar Ξ.

The procedure for computing $\mathbf{Q}_i^{\Xi}$ is demonstrated in Fig. 4(a). For simplicity, we only show the process on the left side of mortar Ξ. To get the solutions, we require that

$$\int_0^1 (\mathbf{Q}^{\Xi,L}(z) - \mathbf{Q}^{\Omega}(\xi)) h_j(z) dz = 0, \quad j = 1, 2, \ldots, N. \tag{20}$$

It was shown in [7] that the above requirement is equivalent to an unweighted L_2 projection. Substituting Eqs. (17)–(19) into the above equation and evaluating it at each SP on Ξ will give a system of linear equations. The solution of this system when written in matrix form is

$$\mathbf{Q}^{\Xi,L} = \mathbf{P}^{\Omega\to\Xi}\mathbf{Q}^{\Omega} = \mathbf{M}^{-1}\mathbf{S}^{\Omega\to\Xi}\mathbf{Q}^{\Omega}, \tag{21}$$

where $\mathbf{P}^{\Omega\to\Xi}$ is the projection matrix from Ω to Ξ, and the elements of the matrices $\mathbf{M}$ and $\mathbf{S}^{\Omega\to\Xi}$ are

$$M_{i,j} = \int_0^1 h_i(z) h_j(z) dz, \quad i, j = 1, 2, \ldots, N, \tag{22}$$

$$S_{i,j}^{\Omega\to\Xi} = \int_0^1 h_i(o + sz) h_j(z) dz, \quad i, j = 1, 2, \ldots, N, \tag{23}$$

where o and s are the offset and the scaling of Ξ with respect to Ω. It is important to note that o and s are time-dependent for the sliding-mesh interface method.

The right solution vector $\mathbf{Q}^{\Xi,R}$ can be computed in the same way. Having both the left and the right solutions on a mortar, the Rusanov solver is employed to compute the common inviscid flux $\mathbf{F}^{\Xi}_{inv}$. This flux is then transformed to the computational flux as $\widetilde{\mathbf{F}}^{\Xi}_{inv}$ according to Eq. (9).

As shown in Fig. 4(b), to project the common inviscid fluxes $\widetilde{\mathbf{F}}^{\Xi_1}_{inv}$ and $\widetilde{\mathbf{F}}^{\Xi_2}_{inv}$ back to face Ω, we require that,

$$\int_0^{o_2} (\widetilde{\mathbf{F}}^{\Omega}_{inv}(\xi) - \widetilde{\mathbf{F}}^{\Xi_1}_{inv}(z))h_j(\xi)d\xi + \int_{o_2}^{1} (\widetilde{\mathbf{F}}^{\Omega}_{inv}(\xi) - \widetilde{\mathbf{F}}^{\Xi_2}_{inv}(z))h_j(\xi)d\xi = 0, \quad j = 1, 2, \ldots, N, \tag{24}$$

where $\widetilde{\mathbf{F}}^{\Omega}_{inv}(\xi)$ is the inviscid flux polynomial on face Ω. The solution of the above equation when written in matrix form is

$$\widetilde{\mathbf{F}}^{\Omega}_{inv} = \mathbf{P}^{\Xi_1\to\Omega}\widetilde{\mathbf{F}}^{\Xi_1}_{inv} + \mathbf{P}^{\Xi_2\to\Omega}\widetilde{\mathbf{F}}^{\Xi_2}_{inv} = s_1\mathbf{M}^{-1}\mathbf{S}^{\Xi_1\to\Omega}\widetilde{\mathbf{F}}^{\Xi_1}_{inv} + s_2\mathbf{M}^{-1}\mathbf{S}^{\Xi_2\to\Omega}\widetilde{\mathbf{F}}^{\Xi_2}_{inv}, \tag{25}$$

where the matrix $\mathbf{M}$ is identical to that of Eq. (21), the matrices $\mathbf{S}^{\Xi_1\to\Omega}$ and $\mathbf{S}^{\Xi_2\to\Omega}$ are simply the transposes of $\mathbf{S}^{\Omega\to\Xi_1}$ and $\mathbf{S}^{\Omega\to\Xi_2}$, respectively.

For the computation of the common viscous fluxes, we first compute the common solution on each mortar as the average of the left and the right solutions,

$$\mathbf{Q}^{\Xi} = \frac{1}{2}(\mathbf{Q}^{\Xi,L} + \mathbf{Q}^{\Xi,R}). \tag{26}$$

This common solution is then projected back to the cell faces in the same procedure as for the inviscid flux in Eq. (25). After that, solution gradients and viscous fluxes are updated on the cell faces on both sides of the interface. The viscous fluxes $\widetilde{\mathbf{F}}^{\Omega}_{vis}$ on the cell faces are projected to the mortars in the same way as Eq. (21). The common viscous flux $\widetilde{\mathbf{F}}^{\Xi}_{vis}$ on a mortar is taken as the average of the left and the right viscous fluxes,

$$\widetilde{\mathbf{F}}^{\Xi}_{vis} = \frac{1}{2}(\widetilde{\mathbf{F}}^{\Xi,L}_{vis} + \widetilde{\mathbf{F}}^{\Xi,R}_{vis}). \tag{27}$$

The final step is to project $\widetilde{\mathbf{F}}^{\Xi}_{vis}$ back to cell faces, which is identical to the process showed in Eq. (25).

Since a uniform mesh is used for the cell faces on the sliding-mesh interface, the matrix $\mathbf{S}$ only needs to be computed for the first two mortars, and can be reused by other corresponding mortars. At the same time since the matrix $\mathbf{M}$ is time independent, it can be computed and stored before the actual calculation. To compute the integrals in Eqs. (22) and (23), one can use the Clenshaw–Curtis quadrature method as was used in [7]. In this study, the integrand is casted into a general form as a product of $2(N-1)$ first degree polynomials, and we implement a recursive algorithm to compute the integrals analytically. This approach requires the least number of operations which is much more efficient than the Clenshaw–Curtis quadrature method.

4. Numerical tests

In this section we test the spatial accuracy of the SSD method on both inviscid and viscous flows, and then apply this method to study an external and an internal flows. A five-stage fourth-order Runge–Kutta method for time stepping [27] is used for all test cases. In each test case for demonstrating the orders of accuracy of spatial discretizations, the time step size is reduced successively until the final errors do not change with it. This ensures that the temporal discretization errors are negligible and the final errors can represent the spatial discretization errors.

4.1. Euler vortex flow

Euler vortex flow has been widely used to test the accuracies of inviscid flow solvers [3,31]. In this problem, an isentropic vortex is superimposed to and convected by a uniform mean flow. The Euler vortex flow in an infinite domain at time t can be analytically described as

$$u = U_\infty\left\{\cos\theta - \frac{\epsilon y_r}{r_c}\exp\left(\frac{1-x_r^2-y_r^2}{2r_c^2}\right)\right\}, \tag{28}$$

$$v = U_\infty\left\{\sin\theta + \frac{\epsilon x_r}{r_c}\exp\left(\frac{1-x_r^2-y_r^2}{2r_c^2}\right)\right\}, \tag{29}$$

$$\rho = \rho_\infty\left\{1 - \frac{(\gamma-1)(\epsilon M_\infty)^2}{2}\exp\left(\frac{1-x_r^2-y_r^2}{r_c^2}\right)\right\}^{\frac{1}{\gamma-1}}, \tag{30}$$

$$p = p_\infty\left\{1 - \frac{(\gamma-1)(\epsilon M_\infty)^2}{2}\exp\left(\frac{1-x_r^2-y_r^2}{r_c^2}\right)\right\}^{\frac{\gamma}{\gamma-1}}, \tag{31}$$

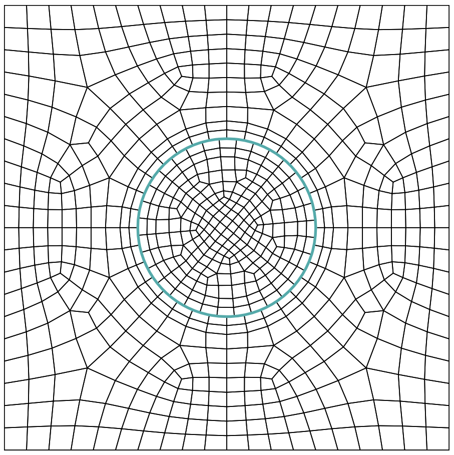

Fig. 5. Mesh with 700 cells at a time instant for the Euler vortex flow simulation (blue circle indicates sliding-mesh interface). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where U_∞, ρ_∞, p_∞, M_∞ are the mean flow speed, density, pressure and Mach number, respectively. θ is the direction of the mean flow (i.e. the direction along which the vortex is convected), ϵ and r_c can be interpreted as the vortex strength and size. The relative coordinates (x_r, y_r) are defined as

$$x_r = x - x_0 - \bar{u}t, \tag{32}$$

$$y_r = y - y_0 - \bar{v}t, \tag{33}$$

where $\bar{u} = U_\infty \cos\theta$, $\bar{v} = U_\infty \sin\theta$ are the x and y components of the mean velocity, (x_0, y_0) is the initial position of the vortex. The analytical solution of the Euler vortex flow problem within a square domain $(0 \le x, y \le L)$ with periodic boundary conditions can be achieved by replacing the relative coordinates with the following expressions,

$$x_r = x_r - \lfloor \frac{x_r + x_0}{L} \rfloor \cdot L, \tag{34}$$

$$y_r = y_r - \lfloor \frac{y_r + y_0}{L} \rfloor \cdot L, \tag{35}$$

where the floor operator $\lfloor x \rfloor$ gives the largest integer that is not greater than a real number x. The x_r and y_r on the right hand sides of Eqs. (34) and (35) are from Eqs. (32) and (33).

In this test, the uniform mean flow is chosen as $(U_\infty, \rho_\infty, p_\infty) = (1, 1, 1)$ with a Mach number of $M_\infty = 0.3$. The flow direction is set to $\theta = \arctan(1/2)$. A vortex with parameters: $\epsilon = 1$, $r_c = 1$, is superimposed to the mean flow. The domain size is $0 \le x, y \le 10$ (i.e. $L = 10$), and the vortex is initially located at the domain center $(x_0, y_0) = (5, 5)$. Periodic boundary conditions are applied in both x and y directions.

Fig. 5 shows a computational mesh with 700 cells. The mesh has been decomposed into two parts: a rotating inner part with a radius of 2; a fixed outer part which takes the rest of the computational domain. Three meshes with 180, 700 and 2731 cells have been used for accuracy tests. For all three cases, the inner part is set to rotate at an angular speed of $\omega = 1.0$.

Fig. 6 compares the density contours obtained with the fourth-oder SSD method on the finest mesh with the exact solution at $t = 2$. As we can see, the solver resolves the vortex very well, and we see almost no visible difference between the exact solution and the numerical one.

Furthermore, Tables 1 and 2 give the spatial accuracy of the scheme, where the $L1$ and $L2$ errors are computed from density at $t = 2$ when vortex center is traveled right onto the sliding-mesh interface. From the two tables we see that the SSD method gives very reasonable order of accuracy.

To see how efficient the SSD method is, we compare the total computational time and the communication time on the sliding-mesh interface in Tables 3 and 4 for the third- and fourth-order schemes, respectively. Times in both tables are collected for 100 computational steps and do not include any post-processing time. It is seen that for all test cases, communication on the sliding-mesh interface takes only a few percent of the total computational time, which clearly shows that the SSD method is efficient. It is interesting that the relative communication time (represented by the percentage) decreases as either the number of cells or the order of schemes increases. This is due to the fact that cells are one dimension higher than faces: when we perform a mesh refinement, the total number of faces in the domain grows faster than on the sliding-mesh interface; when we increase the scheme order, the total number of degrees of freedom in the domain also grows faster than on the sliding-mesh interface.

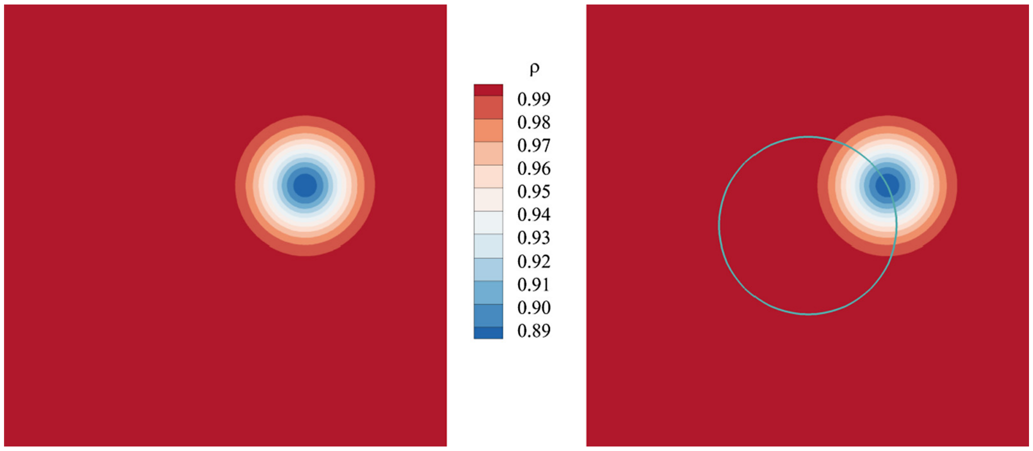

Fig. 6. Contours of density at the time instant $t = 2$ for the Euler vortex flow. Left, exact solution; right, numerical solution from fourth-order scheme (blue circle indicates location of sliding interface). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Errors and orders of accuracy of the third-order scheme for the Euler vortex flow simulation.

cells	$L1$ error	order	$L2$ error	order
180	3.16E−4	–	8.13E−4	–
700	4.58E−5	2.85	1.13E−4	2.90
2731	7.00E−6	2.80	1.73E−5	2.83

Table 2
Errors and orders of accuracy of the fourth-order scheme for the Euler vortex flow simulation.

cells	$L1$ error	order	$L2$ error	order
180	5.43E−5	–	1.26E−4	–
700	3.27E−6	4.14	8.02E−6	4.06
2731	2.26E−7	4.03	5.50E−7	4.00

Table 3
Total computation time and interface communication time (both in seconds) for 100 computational steps using a third-order scheme for the Euler vortex flow simulation.

cells	total time	comm. time	percentage
180	0.254944	0.017657	6.92%
700	1.013039	0.041758	4.12%
2731	4.607205	0.097717	2.12%

Table 4
Total computation time and interface communication time (both in seconds) for 100 computational steps using a fourth-order scheme for the Euler vortex flow simulation.

cells	total time	comm. time	percentage
180	0.420763	0.023282	5.53%
700	1.763656	0.058833	3.34%
2731	7.511551	0.125820	1.68%

4.2. Taylor–Couette flow

To test the order accuracy on viscous flow, we use the laminar Taylor–Couette flow as the test case. Previous researchers such as Liang et al. [13], Michalak and Ollivier-Gooch [21] used similar flows to test the accuracy of their solvers. In the present test, the inner cylinder has a radius of $r_i = 1$, the outer cylinder has radius of $r_o = 2$. Both boundaries are set to be isothermal walls. The domain has been divided into two parts at $r = 1.5$. The inner part rotates at an angular speed of $\omega_i = 1$ while the outer part stays stationary. The Reynolds number based on the inner cylinder radius and speed is $Re = 10$. The Mach number on the inner wall is set to $M_i = 0.1$. Three meshes with 192, 768 and 3072 cells are used for the tests. Fig. 7 shows the mesh with 192 cells.

Fig. 8 shows the steady state contours of the u velocity component and the Mach number obtained with a fourth-oder scheme on the finest mesh. We see that the Mach contours at the steady state are a series of concentric circles, and the u velocity is highly symmetric. These results are consistent with our expectations.

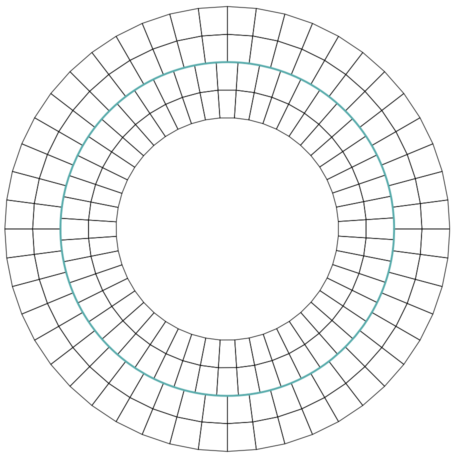

Fig. 7. Mesh with 192 cells at a time instant for the Taylor–Couette flow simulation (blue circle indicates sliding-mesh interface). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

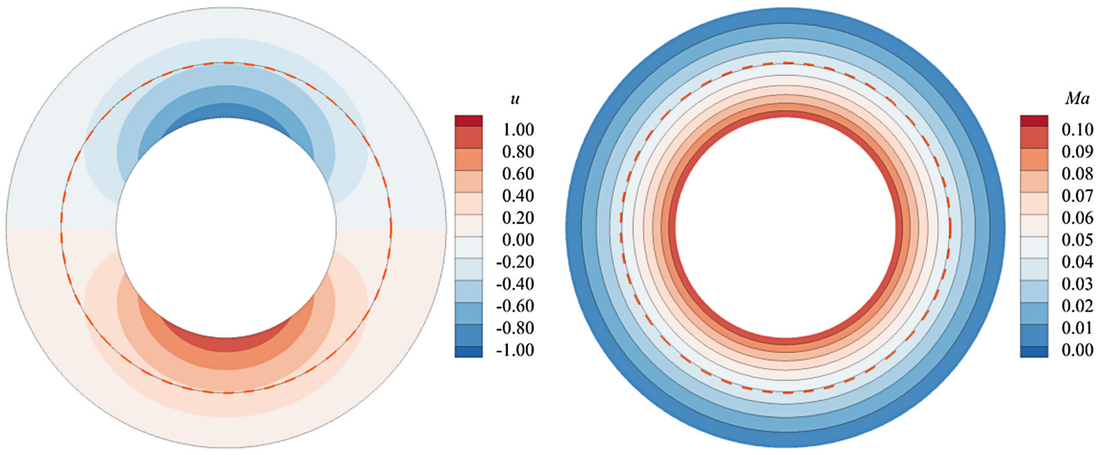

Fig. 8. Contours of u velocity component and the Mach number for the Taylor–Couette flow (dashed circle indicates location of sliding-mesh interface).

Table 5
Errors and orders of accuracy of a third-order scheme for the Taylor–Couette flow simulation.

cells	$L1$ error	order	$L2$ error	order
192	5.90E−5	–	8.71E−5	–
768	6.95E−6	3.09	9.82E−6	3.15
3072	8.43E−7	3.07	1.09E−6	3.16

Table 6
Errors and orders of accuracy of a fourth-order scheme for the Taylor–Couette flow simulation.

cells	$L1$ error	order	$L2$ error	order
192	9.72E−6	–	1.52E−5	–
768	6.16E−7	3.98	1.01E−6	3.91
3072	4.22E−8	3.92	6.60E−8	3.92

The exact solution for the circumferential velocity has the following relation to radius r,

$$v_\theta = \omega_i r_i \frac{r_o/r - r/r_o}{r_o/r_i - r_i/r_o}. \tag{36}$$

The x component of this velocity (i.e., u) is used to compute the $L1$ and $L2$ error norms. From Table 5 and Table 6 we see that the SSD method preserves the high-order accuracy for viscous flow as well.

The total computational time and the sliding-mesh interface communication time are shown in Table 7 and Table 8 for third- and fourth-order schemes, respectively. Again, data in both tables are collected for 100 computational steps and do not include any post-processing time. It is seen that for viscous flow simulations the SSD method remains very efficient.

Table 7
Total computation time and interface communication time (both in seconds) for 100 computational steps using a third-order scheme for the Taylor–Couette flow simulation.

cells	total time	comm. time	percentage
192	1.267700	0.105698	8.34%
768	4.394639	0.234498	5.34%
3072	18.138023	0.561518	3.10%

Table 8
Total computation time and interface communication time (both in seconds) for 100 computational steps using a fourth-order scheme for the Taylor–Couette flow simulation.

cells	total time	comm. time	percentage
192	2.075053	0.139513	6.92%
768	7.628162	0.322616	4.12%
3072	30.852314	0.729768	2.12%

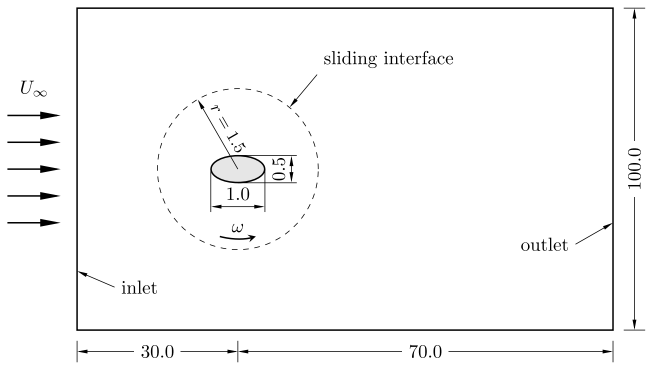

Fig. 9. Schematic of the computational domain for flow over a rotating elliptic cylinder (not to scale).

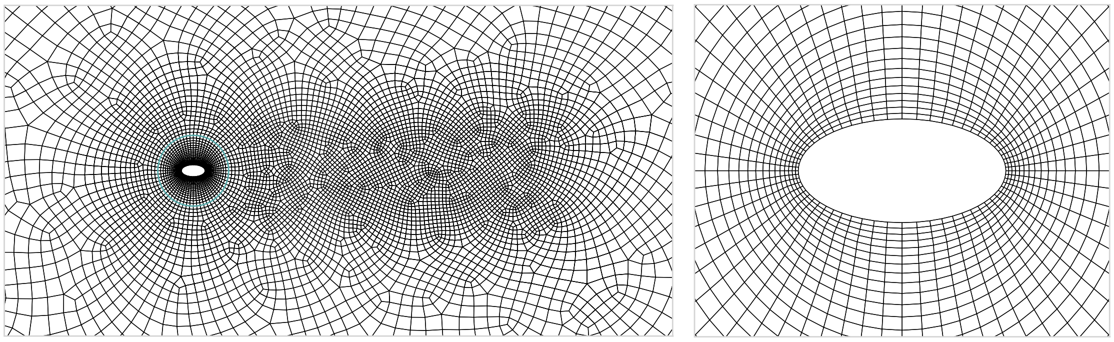

Fig. 10. Two local views of the mesh for simulation of flow around a rotating elliptic cylinder (blue circle indicates sliding-mesh interface). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.3. Flow over a rotating elliptic cylinder

To further verify the approach for flow involving complex geometries, we simulate a laminar flow over a two-dimensional elliptic cylinder in this section. Maruoka [17] and Zhang et al. [35] studied incompressible flow over a rotating elliptic cylinder using the finite element and finite volume methods, respectively. Both studies use Chimera grids for communication between the foreground rotating mesh and the background stationary mesh. The freestream Mach number in the present test is set to $M_\infty = 0.05$ to mitigate the compressibility effects in order to compare with their incompressible flow results.

The elliptic cylinder has a major and a minor axis lengths of 1.0 and 0.5, respectively. Initially, the major axis is parallel to the freestream. The cylinder rotates counterclockwisely at an angular speed of $\omega = 0.5\pi$. The Reynolds number based on the freestream velocity and the major axis length is $Re = 200$. Fig. 9 shows a schematic of the computational domain. Slip boundary conditions are applied on the top and bottom boundaries. Dirichlet boundary condition is used at the inlet, and fixed pressure boundary condition is applied at the outlet. Finally, no-slip isothermal wall boundary condition is used on the cylinder surface.

The inner rotating domain has a radius of 1.5 and is meshed with 1280 cells. The rest of the domain is stationary and has 7391 cells. Mesh refinement is performed around the leading and the trailing edges, as well as in the wake region. Fig. 10

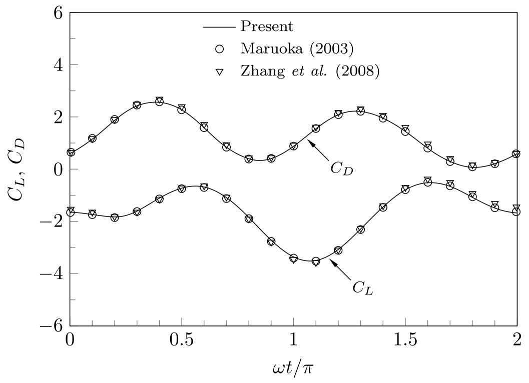

Fig. 11. Lift and drag coefficients for flow over an elliptic cylinder.

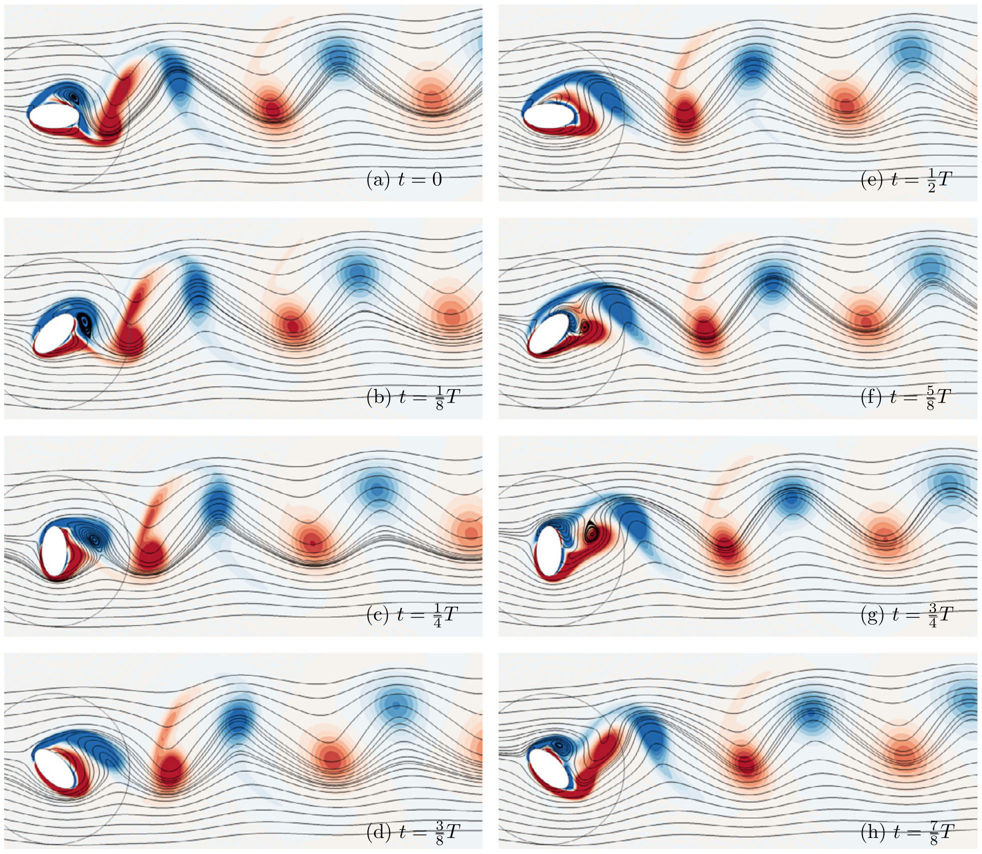

Fig. 12. Streamlines and vorticity contours (blue color means negative value, red color means positive) for flow over a rotating elliptic cylinder. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

shows two local views of the mesh. The nondimensional time step size $\Delta t U_\infty/L$ for the simulation is set to 1.0×10^{-4}, where L is the major axis length and U_∞ is the freestream velocity.

Both third- and fourth-order schemes were tested for this flow and no visible difference was observed between the solutions. For this reason, we only present results from the fourth-order scheme. As was noticed by Maruoka [17] and Zhang et al. [35], the fully developed flow takes a periodic pattern as the cylinder rotates. The lift and the drag coefficients in one rotating period are shown in Fig. 11. It is seen that the present results agree very well with the previously published results.

Fig. 12 shows the streamlines superimposed on vorticity contours at a series of time instants in one rotating period. A clockwise and a counterclockwise vortices appear alternatively around the two ends of the cylinder. From Fig. 12(h) and Fig. 12(a), we see that a clockwise vortex is formed at the cylinder leading edge as the cylinder rotates, and this vortex is then shed off from the leading edge and travels downstream to hit the trailing edge. From Fig. 12(b) to Fig. 12(d), the

Table 9
Total computation time and interface communication time (both in seconds) for 100 computational steps for simulation of flow over a rotating elliptic cylinder.

order	total time	comm. time	percentage
3	38.924785	0.143083	0.37%
4	70.655175	0.185890	0.26%

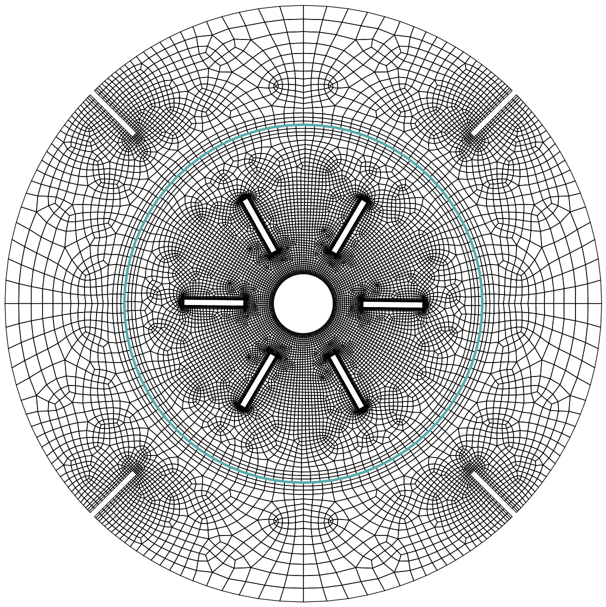

Fig. 13. Mesh for simulation of flow inside a 2*D* stirred tank (blue circle indicates sliding-mesh interface). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

same clockwise vortex is again shed off from the trailing edge and then convected towards downstream. From Fig. 12(e) to Fig. 12(g), a counterclockwise vortex slowly emerges around the other end of the cylinder and is then convected downstream without reattaching to the cylinder. This process repeats as the cylinder rotates, and a vortex street forms downstream of the cylinder.

The efficiency of the SSD method is shown in Table 9 for this case. The results in the table confirm our previous conclusion, i.e., the relative communication time generally decreases as the number of cells or the order of scheme increases. In fact, the interface communication time in the table is almost negligible comparing to the total computational time.

4.4. Flow inside a 2D stirred tank

In this last test case, we apply the SSD method to simulate laminar flow inside a 2*D* stirred tank. Fig. 13 shows the unstructured quadrilateral mesh used for this case. As we can see, the tank has several components: an inner circular wall with a radius of 0.5; an outer circular wall with a radius of 5; six uniformly distributed agitating blades, each extends from $r = 1$ to $r = 2$ and has a thickness of 0.1; four baffles installed on the outer wall, each of them has a height of 1 and a thickness of 0.1. The computational domain is split into an inner rotating part and an outer fixed part, resulting in a sliding-mesh interface at $r = 3$. Mesh has been refined around the blades, baffles, and the wall boundaries. The resulted mesh has 14,990 cells in total.

In this simulation, the inner circular wall and the blades rotate at an angular speed of $\omega = 1$. The Reynolds number based on the inner wall diameter and the angular speed is $Re = 100$. Flow on the inner wall surface has a Mach number of $M_i = 0.1$. No slip isothermal wall boundary conditions are used on the inner wall surface, the baffles, and the outer walls. Adiabatic wall boundary conditions are used on the six blades. A time step size of $\Delta t = 1.0 \times 10^{-4}$ is used.

Initially, the flow field is set to be uniform and stationary. Fig. 14 shows the flow fields at four different states by visualizing the density. It is seen in Fig. 14(a) that at the initial transient state, the flow behaves like flow around a propeller: fluid is “squeezed” and “pushed” away from the blades, resulting in a concentric flow pattern, and large fluctuations are observed in the flow field. Quickly after the flow reaches the baffles and the outer wall, the flow field becomes very chaotic: vortical structures generated by flow passing the baffles, bouncing pressure waves, blades induced vortices, unsteady boundary layers, etc., as can be seen in Fig. 14(b). As the blades continue rotating for a longer time, the chaotic flow structures are slowly being dissipated, and organized large flow structures emerge as shown in Fig. 14(c). Finally the flow field reaches a quasi steady state in the rotating reference frame, carrying little variation with time, as shown in Fig. 14(d).

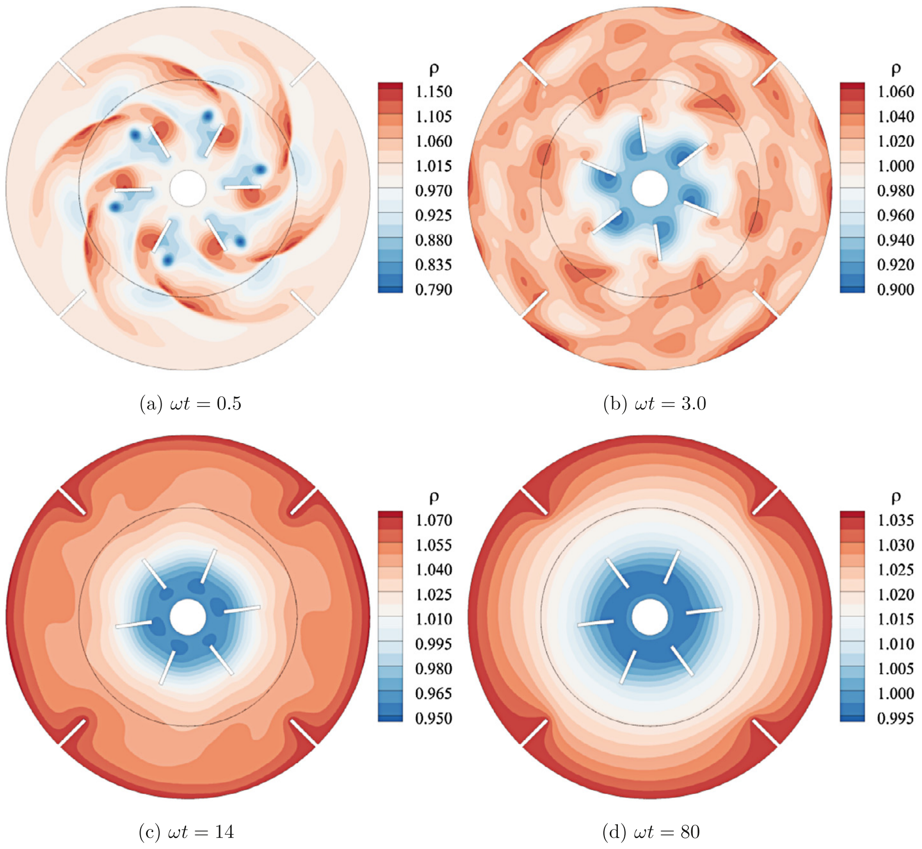

Fig. 14. Density contours of flow inside a $2D$ stirred tank at four distinct states.

Table 10
Total computation time and interface communication time (both in seconds) for 100 computational steps for simulation of flow inside a $2D$ stirred tank.

order	total time	comm. time	percentage
3	85.547403	0.328987	0.38%
4	148.780491	0.437902	0.29%

Interestingly, the flow is becoming more and more uniform at the later stage of stirring. Variations on the density field get smaller and smaller when comparing Fig. 14(a)–(d) along time.

As for all previous cases, we monitored the efficiency of the SSD method for this case. The results are shown in Table 10. Again, as can be seen from the table, the SSD method is very efficient with negligible communication time.

5. Conclusions

In this paper, a novel, simple, efficient, and high-order accurate sliding-mesh interface method is reported for subsonic compressible flows. The sliding-mesh spectral difference (SSD) method has been successfully developed and tested for several inviscid and viscous flow problems. The SSD method retains the high-order accuracy of the SD method. It is also shown that the SSD method is very efficient as it introduces negligible extra computational cost to the SD method for realistic flow simulations. In this paper, we demonstrated the approach on uniformly meshed sliding-mesh interfaces where each cell face has two mortars. The sliding-mesh interface method can be extended to handle more than two mortars for each face. The SSD method is also very suitable for parallel computing. Since there is no overlapping in grids and the sliding-mesh interface method introduces negligible computational time, each domain can be decomposed and distributed to the processors to achieve load balancing. Finally, this high-order curved sliding-mesh interface method can also be extended to other discontinuous high-order methods for compressible flows.

Acknowledgements

The authors would like to express their acknowledgments for an Office of Naval Research (ONR) support with award number N000141210500 managed by Dr. Ki-Han Kim. Chunlei Liang would also like to acknowledge the support by an ONR Young Investigator Program award administrated by Dr. Ki-Han Kim.

References

[1] P. Castonguay, C. Liang, A. Jameson, Simulation of transitional flow over airfoils using the spectral difference method, AIAA paper 2010-4626, 2010.
[2] A. DeJong, C. Liang, Parallel spectral difference method for predicting 3D vortex-induced vibrations, Comput. Fluids 98 (2014) 17–26.
[3] G. Erlebacher, M.Y. Hussaini, C.-W. Shu, Interaction of a shock with a longitudinal vortex, J. Fluid Mech. 337 (1997).
[4] C.W. Hirt, A.A. Amsden, J.L. Cook, An arbitrary Lagrangian–Eulerian computing method for all flow speeds, J. Comput. Phys. 14 (1974) 227–253.
[5] H. Huynh, Z. Wang, P. Vincent, High-order methods for computational fluid dynamics: a brief review of compact differential formulations on unstructured grids, Comput. Fluids 98 (2014) 209–220.
[6] A. Jameson, A proof of the stability of the spectral difference method for all orders of accuracy, J. Sci. Comput. 45 (2010) 348–358.
[7] D.A. Kopriva, A conservative staggered-grid Chebyshev multidomain method for compressible flows. II. A semi-structured method, J. Comput. Phys. 128 (1996) 475–488.
[8] D.A. Kopriva, A staggered-grid multidomain spectral method for the compressible Navier–Stokes equations, J. Comput. Phys. 143 (1998) 125–158.
[9] D.A. Kopriva, J.H. Kolias, A conservative staggered-grid Chebyshev multidomain method for compressible flows, J. Comput. Phys. 125 (1996) 244–261.
[10] C. Liang, A. Jameson, Z.J. Wang, Spectral difference method for two-dimensional compressible flow on unstructured grids with mixed elements, J. Comput. Phys. 228 (2009) 2847–2858.
[11] C. Liang, R. Kannan, Z.J. Wang, A p-multigrid spectral difference method with explicit and implicit smoothers on unstructured triangular grids, Comput. Fluids 38 (2009) 254–265.
[12] C. Liang, K. Ou, S. Premasuthan, A. Jameson, Z. Wang, High-order accurate simulations of unsteady flow past plunging and pitching airfoils, Comput. Fluids 40 (2011) 236–248.
[13] C. Liang, S. Premasuthan, A. Jameson, High-order accurate simulation of low-Mach laminar flow past two side-by-side cylinders using spectral difference method, Comput. Struct. 87 (2009) 812–817.
[14] C. Liang, S. Premasuthan, A. Jameson, Z.J. Wang, Large eddy simulation of compressible turbulent channel flow with spectral difference method, AIAA paper 2009-2402, 2009.
[15] Y. Liu, M. Vinokur, Z.J. Wang, Spectral difference method for unstructured grids I: basic formulation, J. Comput. Phys. 216 (2006) 780–801.
[16] G. Lodato, P. Castonguay, A. Jameson, Structural wall-modeled LES using a high-order spectral difference scheme for unstructured meshes, Flow Turbul. Combust. 92 (1–2) (2014) 579–606.
[17] A. Maruoka, Finite element analysis for flow around a rotating body using Chimera method, Int. J. Comput. Fluid Dyn. 17 (4) (2003) 289–297.
[18] C.A. Mavriplis, Nonconforming discretizations and a posteriori error estimators for adaptive spectral element techniques, Ph.D. thesis, M.I.T., February 1989.
[19] G. May, On the connection between the spectral difference method and the discontinuous Galerkin method, Commun. Comput. Phys. 9 (2011) 1071–1080.
[20] J. McNaughton, I. Afgan, D.D. Apsley, S. Rolfo, T. Stallard, P.K. Stansby, A simple sliding-mesh interface procedure and its application to the CFD simulation of a tidal-stream turbine, Int. J. Numer. Methods Fluids 74 (2014) 250–269.
[21] C. Michalak, C. Ollivier-Gooch, Unstructured high-order accurate finite-volume solutions of the Navier–Stokes equations, AIAA paper 2009-2954, 2009.
[22] A.H. Mohammad, Z.J. Wang, C. Liang, LES of turbulent flow past a cylinder using spectral difference method, Adv. Appl. Math. Mech. 2 (2010) 451–466.
[23] K. Ou, C. Liang, A. Jameson, High-order spectral difference method for the Navier–Stokes equations on unstructured moving deforming grids, AIAA paper 2010-0541, 2010.
[24] M. Parsani, G. Ghorbaniasl, C. Lacor, Validation and application of an high-order spectral difference method for flow induced noise simulation, J. Comput. Acoust. 19 (3) (2011) 241–268.
[25] M. Parsani, G. Ghorbaniasl, C. Lacor, E. Turkel, An implicit high-order spectral difference approach for large eddy simulation, J. Comput. Phys. 229 (2010) 5373–5393.
[26] V.V. Rusanov, Calculation of interaction of non-steady shock waves with obstacles, J. Comput. Math. Phys. USSR 1 (1961) 267–279.
[27] R.J. Spiteri, S.J. Ruuth, A new class of optimal high-order strong-stability-preserving time discretization methods, SIAM J. Numer. Anal. 40 (2002) 469–491.
[28] Y. Sun, Z.J. Wang, Y. Liu, High-order multidomain spectral difference method for the Navier–Stokes equations on unstructured hexahedral grids, Commun. Comput. Phys. 2 (2007) 310–333.
[29] T.E. Tezduyar, M. Behr, A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space-time procedure: I. The concept and the preliminary numerical tests, Comput. Methods Appl. Mech. Eng. 94 (3) (February 1992) 339–351.
[30] T.E. Tezduyar, M. Behr, S. Mittal, A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space-time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders, Comput. Methods Appl. Mech. Eng. 94 (3) (February 1992) 353–371.
[31] Z.J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H.T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, M. Visbal, High-order CFD methods: current status and perspective, Int. J. Numer. Methods Fluids 72 (8) (2013) 811–845.
[32] Z.J. Wang, Y. Liu, G. May, A. Jameson, Spectral difference method for unstructured grids II: extension to the Euler equations, J. Sci. Comput. 32 (2007) 45–71.
[33] S.L. Yeoh, G. Papadakis, K. Lee, M. Yianneskis, Large eddy simulation of turbulent flow in a Ruston impeller stirred reactor with sliding-deforming mesh methodology, Chem. Eng. Technol. 27 (2004) 258–263.
[34] M.L. Yu, Z.J. Wang, H. Hu, A high-order spectral difference method for unstructured dynamic grids, Comput. Fluids 48 (2011) 84–97.
[35] X. Zhang, S. Ni, G. He, A pressure-correction method and its applications on an unstructured Chimera grid, Comput. Fluids 37 (2008) 993–1010.