A high order moving boundary treatment for convection-diffusion equations

Shihao Liu [1], Yan Jiang [2], Chi-Wang Shu[3], Mengping Zhang[4] and Shuhai Zhang[5]

Abstract

In this paper, a new type of inverse Lax-Wendroff boundary treatment is designed for high order finite difference schemes for solving general convection-diffusion equations on time-varying domain. This new method can achieve high order accuracy on Dirichlet boundary conditions with moving boundary. To ensure stability of the boundary treatment, we give a convex combination of the boundary treatments for the diffusion-dominated and the convection-dominated cases. A group convex combination of weights is carefully designed to avoid zero denominator, resulting in a unified algorithm for pure convection, convection-dominated, convection-diffusion, diffusion-dominated and pure diffusion cases. In order to match the time levels when constructing values of ghost points in the two intermediate stages of the third order Runge-Kutta method, we propose a new approximation to the mixed derivatives at the boundaries to ensure high order accuracy and to improve computational efficiency. In particular, we extend the boundary treatment to the compressible Navier-Stokes equations, which satisfies the isothermal no-slip wall boundary condition at any Reynolds number. We provide numerical tests for one- and two-dimensional problems involving both scalar equations and systems, demonstrating that our boundary treatment is high order accurate for problems with smooth solutions and also performs well for problems involving interactions between viscous shocks and moving rigid bodies.

Key Words: convection-diffusion equations, compressible Navier-Stokes equations, complex moving boundaries, numerical boundary conditions, inverse Lax-Wendroff method, Cartesian mesh.

[1]School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui 230026, China. E-mail: liu0919@mail.ustc.edu.cn.

[2]School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui 230026, China. E-mail: jiangy@ustc.edu.cn. Research is supported in part by NSFC grant 11901555.

[3]Division of Applied Mathematics, Brown University, Providence, RI 02912, USA. E-mail: chi-wang_shu@brown.edu. Research is supported in part by AFOSR grant FA9550-20-1-0055 and NSF grant DMS-2010107.

[4]School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui 230026, China. E-mail: mpzhang@ustc.edu.cn. Research is supported in part by National Numerical Windtunnel project NNW2019ZT4-B10 and NSFC grant 11871448.

[5]State Key Laboratory of Aerodynamics, China Aerodynamics Research and Development Center, Mianyang, Sichuan 621000, China. E-mail: shuhai_zhang@163.com. Research is supported in part by NSFC grant 11732016.

1 Introduction

In this paper, we develop a high order accurate numerical boundary treatment based on finite difference methods for convection-diffusion equations on complex moving domain. For such complex geometries, there are two main difficulties for numerical boundary conditions: one is the wide stencil of the high order finite difference operator, which requires special treatment for a few ghost points near the boundary. The other one is that the physical boundary may not coincide with grid points in a Cartesian mesh and may intersect with the mesh in an arbitrary fashion, then the so-called "cut-cell" problem [2] may arise, which means that extremely small CFL condition may be required when the first grid point does not coincide with but is very close to the physical boundary. The body-fitted grid method can be very accurate for the treatment of boundary conditions in complex geometries. However, dealing with a moving domain, the mesh needs to be computed and regenerated at each time step, resulting in a heavy workload. For a non-body fitted Cartesian (Eulerian) mesh, there are many carefully designed methods, such as the embedded boundary method [9, 10, 8, 17, 21, 1], and the immersed boundary method [18, 16, 19, 26, 28].

In this paper, we will focus on the so-called inverse Lax-Wendroff (ILW) method. This method was first proposed by Tan and Shu in [22] to deal with the inflow boundary conditions when solving time-dependent hyperbolic conservation laws. The key idea of the ILW method is repeatedly utilizing the partial differential equations (PDEs) to write the normal spatial derivatives to the inflow boundary in terms of the time and tangential derivatives of the given boundary condition. At outflow boundaries, a weighted essentially non-oscillatory (WENO) type extrapolation was designed for the case of shock waves being close to the boundaries. Then, with these normal derivatives, ghost point values near the physical boundary can be obtained through Taylor expansions. Due to the heavy algebra of the ILW procedure for high dimensional nonlinear systems, [25] developed a simplified ILW (SILW) procedure to reduce the computation complexity, in which the ILW process is only used for the first few normal derivatives and the less expensive high order extrapolation is used for the remaining ones. Stability analysis for both the ILW and the simplified ILW procedures is given in [27, 11], indicating that the ILW and the simplified ILW procedure can remove the “cut cell” problem effectively. Recently, [5] modified the numerical fluxes near boundaries to achieve mass conservation in non-rectangular domains, and [15] defined the unknown variables and the fluxes on the ghost points separately such that it can deal with hyperbolic conservation laws with changing wind direction on the boundary. Besides, this procedure and analysis have been explored and developed further to other equations, such as the Boltzmann type equation [6] and convection-diffusion equations [14, 12, 13]. This (S)ILW procedure can be extended to solve the compressible inviscid Euler equations on the moving geometries [23, 4]. In this paper, we would like to design numerical boundary conditions for convection-diffusion equations on a time-varying domain.

We will follow the idea in [14], in which a careful convex combination of the boundary treatments for the diffusion-dominated and the convection-dominated cases was proposed to obtain a stable and accurate boundary condition for high order finite difference schemes when applied to convection-diffusion equations on the static domain, regardless of the regimes. When extending it to the time-varying domain in this paper, there are two underlying issues. Firstly, since conditions are defined on the moving boundaries, we should use material derivatives in the ILW procedure instead of Eulerian time derivatives as in [14]. Secondly, in coupling with third order Runge-Kutta time discretization, special treatment for boundary values in the two intermediate stages is required to ensure high order accuracy. Moreover, we attempt to give new combination weights to avoid zero denominator, such that the method can be used to deal with all the cases from and including pure convection equations to and including pure diffusion equations.

This paper is organized as follows. In Section 2, we first illustrate our idea by developing our inverse Lax-Wendroff method for one-dimensional scalar convection-diffusion equations with Dirichlet boundary conditions on moving boundaries. We then generalize it to one-dimensional systems and two-dimensional problems. In Section 3, we will apply our method to simulate the interactions between shocks and moving rigid bodies with complex geometries. Numerical results are given in Section 4 to demonstrate the effectiveness and robustness of our approach. Concluding remarks are given in Section 5.

2 Scheme formulation

We consider a convection-diffusion equation, which can be written in the following form:

$$\mathbf{U}_t = \mathcal{L}(\mathbf{U}), \quad \text{in } \Omega(t), \tag{2.1}$$

where, $\mathcal{L}$ is a spatial operator involving convection terms and diffusion terms, and $\Omega(t)$ is the time-varying domain. We use the idea of method of lines with a high-order spatial discretization operator $\mathcal{L}_h$ (in our later computation, we use upwind spatial discretization such as the finite difference WENO [7] scheme for the convection terms and high-order central difference for the diffusion terms), which can approximate the original partial differential equations into the following semi-discrete ODE:

$$\mathbf{U}_t = \mathcal{L}_h(\mathbf{U}). \tag{2.2}$$

We use the following third order total variation diminishing (TVD) Runge-Kutta (RK) method [20] to discretize (2.2):

$$\begin{aligned}
\mathbf{U}^{(1)} &= \mathbf{U}^n + \Delta t \mathcal{L}_h(\mathbf{U}^n), \\
\mathbf{U}^{(2)} &= \frac{3}{4}\mathbf{U}^n + \frac{1}{4}\mathbf{U}^{(1)} + \frac{1}{4}\Delta t \mathcal{L}_h(\mathbf{U}^{(1)}), \\
\mathbf{U}^{n+1} &= \frac{1}{3}\mathbf{U}^n + \frac{2}{3}\mathbf{U}^{(2)} + \frac{2}{3}\Delta t \mathcal{L}_h(\mathbf{U}^{(2)}).
\end{aligned} \tag{2.3}$$

We consider the initial-boundary problems with moving Dirichlet boundary conditions. The remaining issue is only the numerical treatment of boundary conditions. Because a high order finite difference operator with a wide stencil needs special treatment for some ghost points near the boundary, we will design an algorithm to use the information of the partial differential equations and boundary conditions to define the values of the ghost points.

Notice that, when we update values of the interior points from time level t_n to time level t_{n+1} with (2.3), there may be some points inside the domain $\Omega(t_{n+1})$ but outside $\Omega(t_n)$. We call these points "newly emerging points". As in [23], there is no need for any special treatment for these points. We only need to construct one more ghost point in each direction at time level t_n and restrict the time step Δt to make sure that the moving domain $\Omega(t)$ travels at most one grid from time level t_n to t_{n+1}.

In this section, we will begin with the one-dimensional convection-diffusion equations to illustrate the moving boundary treatment, and then extend the algorithm to two-dimensional systems.

2.1 One-dimensional scalar convection-diffusion equations with moving boundaries

We consider the following one-dimensional scalar convection-diffusion equation:

$$\begin{cases} u_t + f(u)_x = \epsilon u_{xx}, \quad x \in \Omega(t) = (x_l(t), x_r(t)),\ t > 0, \\ u(a(t), t) = g_l(t), \quad t > 0, \\ u(b(t), t) = g_r(t), \quad t > 0, \\ u(x, 0) = u_0(x), \quad x \in \Omega(t), \end{cases} \tag{2.4}$$

where $(x_l(t), x_r(t)) = (a(t), b(t))$ are given functions varying with time and $\epsilon > 0$ is a constant.

Assume the domain is divided by the uniform mesh at time level t_n

$$x_{-2} < x_{-1} < a(t_n) < x_0 < \cdots < x_N < b(t_n) < x_{N+1} < x_{N+2}, \tag{2.5}$$

with mesh size Δx. Notice that the physical boundary $x = a(t_n)$ and $x = b(t_n)$ may not coincide with the grid points. We take $\{x_0, x_1, \cdots, x_N\}$ as our interior points. At the interior grid point x_j, we have following semi-discrete approximation of (2.4)

$$\frac{d}{dt}u_j(t) + \frac{1}{\Delta x}\left(\hat{f}_{j+1/2} - \hat{f}_{j-1/2}\right) = diff_j, \quad j = 0, \ldots, N. \tag{2.6}$$

Here, u_j is the numerical approximation to the exact solution u at the grid point (x_j, t). The numerical flux $\hat{f}_{j+1/2}$ can be obtained by any reasonable finite difference scheme, such as the third order finite difference WENO reconstruction [7]. This scheme requires a five-point stencil at each point. The right-hand term $diff_j$ is a fourth order center difference discretization of the diffusion term,

$$diff_j = \frac{\epsilon}{12\Delta x^2}(-u_{j-2} + 16u_{j-1} - 30u_j + 16u_{j+1} - u_{j+2}).$$

Hence, up to two ghost points are needed near the boundaries.

Here, we choose the left moving boundary $x = a(t_n)$ as an example and concentrate on how to define the values at the ghost points $\{x_{-2}, x_{-1}\}$. To construct values of the ghost points at time level t_n, we use a third order Taylor expansion

$$u_j = u^{*(0)} + (x_j - a(t_n))u^{*(1)} + \frac{(x_j - a(t_n))^2}{2}u^{*(2)}, \qquad j = -2, -1, \tag{2.7}$$

where, $u^{*(k)}$ is the numerical approximation of the k-th spatial derivative of u at the left boundary $\partial_x^k u|_{(x=a(t_n), t=t_n)}$ with suitable order of accuracy. Given the PDE and the boundary condition (2.4), it is straightforward to define the point value as

$$u^{*(0)} = u(a(t_n), t_n) = g_l(t_n). \tag{2.8}$$

However, the derivative values can not be obtained directly.

To ensure the stability of the boundary scheme, [14] pointed out that, for the convection-diffusion equation with fixed boundaries, the approximation of the spatial derivatives on the boundary should be a convex combination of those obtained for the convection-dominated case and for the diffusion-dominated case.

Following this line and turning to our moving boundary case, to obtain the derivative values, there are two possible numerical methods. One way to obtain approximations to these spatial derivatives at the boundary is through the traditional Lagrangian or WENO extrapolation [22, 25, 15] from interior points with suitable order of accuracy. We use the subscript "ext" to denote derivatives obtained through the extrapolation procedure. The other way is the so-called inverse Lax-Wendroff (ILW) procedure, that spatial derivatives are obtained from the given boundary condition and the PDE by converting the spatial derivative into the time derivatives:

$$g_l'(t) = \frac{D}{Dt}u(a(t),t) = u_t + a'(t)u_x = (a'(t) - f'(u))u_x + \epsilon u_{xx}. \tag{2.9}$$

Notice that the function $g_l(t)$ is defined on the moving boundary, hence we employ the material derivative instead of the Eulerian time derivative as in [14]. We use the subscript "ilw" to denote derivatives obtained through the ILW procedure.

To describe the construction of $u^{*(1)}$ and $u^{*(2)}$, we start with two extreme cases:

Case 1. $a'(t) - f'(g_l(t)) = 0$. For this case, we can only obtain the second order spatial derivative $u_{xx} = \frac{g_l'(t_n)}{\epsilon}$, and denote this as the $u_{ilw}^{*(2)}$. The first order spatial derivative has to come from the extrapolation $u_{ext}^{*(1)}$.

Case 2. $\epsilon = 0$, i.e., pure convection. If $a'(t) - f'(g_l(t)) > 0$, the left boundary is an outflow boundary, we can use extrapolation to define $u^{*(k)}$ from interior points with suitable order of accuracy. If $a'(t) - f'(g_l(t)) \le 0$, the left boundary is an inflow boundary. Then the boundary condition $u(a(t),t) = g_l(t)$ should be imposed to ensure the well-posedness. Thus,

$$u^{*(1)} = \frac{g_l'(t_n)}{a'(t_n) - f'(g_l(t_n))}. \tag{2.10}$$

Again, we can obtain $u^{*(2)}$ in terms of time derivatives through repeatedly using the PDE. This is the key idea of the inverse Lax-Wendroff procedure. In particular, [11] proved that, if only the solution and first order spatial derivative are obtained via the ILW procedure, and $u^{*(2)}$ is obtained from extrapolation directly, the third order scheme is linearly stable. Hence, the algorithm would be simpler and more efficient. This is the so-called simplified ILW (SILW) method proposed in [24].

In the general case when $a'(t) - f'(g_l(t)) \neq 0$ and $\epsilon \neq 0$, the first derivative and the second derivative are coupled in (2.9). There are two ways to construct the derivatives.

One is obtaining the second order spatial derivative by extrapolation and then obtaining the first order spatial derivative through the ILW procedure based on (2.9):

$$u_{ilw}^{*(1)} = \begin{cases} u_{ext}^{*(1)}, & f'(g_l(t_n)) - a'(t_n) \leq 0, \\ -\dfrac{g_l'(t_n) - \epsilon u_{ext}^{*(2)}}{f'(g_l(t_n)) - a'(t_n)}, & f'(g_l(t_n)) - a'(t_n) > 0. \end{cases} \tag{2.11}$$

This will degenerate to the SILW for the case 2 as $\epsilon = 0$. Thus it is expected to work in the convection-dominated regime. Alternately, we may obtain the first order spatial derivative by extrapolation, then obtain the second order spatial derivative through the ILW procedure

$$u_{ilw}^{*(2)} = \frac{g_l'(t) + (f'(g_l(t)) - a'(t)) \cdot u_{ext}^{*(1)}}{\epsilon}. \tag{2.12}$$

This is similar to what we would do for the case 1 and therefore is expected to work in the diffusion-dominated regime.

Following this line and turning to our moving boundary case, the approximation of the spatial derivatives $u^{*(0)}, u^{*(1)}$ and $u^{*(2)}$ can be respectively defined as

$$\begin{aligned} u^{*(1)} =& \omega_1 \cdot u_{ilw}^{*(1)} + \omega_2 \cdot u_{ext}^{*(1)}, \\ u^{*(2)} =& \omega_1 \cdot u_{ext}^{*(2)} + \omega_2 \cdot u_{ilw}^{*(2)}, \end{aligned} \tag{2.13}$$

with the weights

$$\begin{aligned} \omega_1 =& \frac{(f'(g_l(t_n)) - a'(t_n))^2 \Delta x^2}{(f'(g_l(t_n)) - a'(t_n))^2 \Delta x^2 + 9\epsilon^2}, \\ \omega_2 =& 1 - \omega_1 = \frac{9\epsilon^2}{(f'(g_l(t_n)) - a'(t_n))^2 \Delta x^2 + 9\epsilon^2}. \end{aligned} \tag{2.14}$$

Notice that, if $f'(g_l(t)) - a'(t)$ or ϵ is close to 0, which means the velocity of the moving boundary is close to the slope of the characteristic line (i.e., the speed of wave propagation of pure advection problem) at the boundary or the diffusion is very weak at the boundary, the computation of (2.11) or (2.12) may produce large rounding error or may even blow up. Hence, we replace the term $\omega_1^{(1)} \cdot u_{ilw}^{*(1)}$ in (2.13) by

$$\omega_1 \cdot u_{ilw}^{*(1)} = -\frac{(f'(g_l(t_n)) - a'(t_n)) \Delta x^2}{(f'(g_l(t_n)) - a'(t_n))^2 \Delta x^2 + 9\epsilon^2} \left(g_l'(t_n) - \epsilon u_{ext}^{*(2)} \right) \tag{2.15}$$

for the case $f'(g_l(t_n)) - a'(t_n) > 0$. Also $\omega_2^{(2)} \cdot u_{ilw}^{*(2)}$ can be replaced by:

$$\omega_2 \cdot u_{ilw}^{*(2)} = \frac{9\epsilon}{(f'(g_l(t_n)) - a'(t_n))^2 \Delta x^2 + 9\epsilon^2} \left(g_l'(t_n) + (f'(g_l(t_n)) - a'(t_n)) u_{ext}^{*(1)} \right). \tag{2.16}$$

If $f'(g_l(t_n)) - a'(t_n)$ and ϵ are both close to 0, the computation of the weights (2.14) may produce large rounding error as well. To avoid zero denominator, we change the combination and weights as follows:

$$\begin{aligned} u^{*(1)} &= \omega_1^{(1)} \cdot u_{ilw}^{*(1)} + \omega_2^{(1)} \cdot u_{ext}^{*(1)}, \\ u^{*(2)} &= \omega_1^{(2)} \cdot u_{ext}^{*(2)} + \omega_2^{(2)} \cdot u_{ilw}^{*(2)}, \end{aligned} \tag{2.17}$$

where

$$\begin{aligned} \omega_1^{(1)} &= \frac{(f'(g_l(t_n)) - a'(t_n))^2 \Delta x^2}{(f'(g_l(t_n)) - a'(t_n))^2 \Delta x^2 + (3\epsilon + \mu)^2}, \quad \omega_2^{(1)} = 1 - \omega_1^{(1)}, \\ \omega_2^{(2)} &= \frac{9\epsilon^2}{(f'(g_l(t_n)) - a'(t_n) + \mu)^2 \Delta x^2 + 9\epsilon^2}, \quad \omega_1^{(2)} = 1 - \omega_2^{(2)}. \end{aligned} \tag{2.18}$$

Here, μ can be chosen as any small positive number to avoid possible zero in the denominator for extreme cases, in which the velocity of the moving boundary is close to the slope of the characteristic line at the boundary and the diffusion is very weak at the boundary. When $\mu = 0$, the weights (2.18) will recover those in (2.14). In this paper, we set $\mu = 10^{-6}$ in our later computation such that it could be very close to the weights in (2.14) for general cases. Correspondingly, denominators in (2.15) and (2.16) should be changed as well. As a consequence, our method can be applied to solve the pure convection equations and pure diffusion equations, and everything in between. This will be verified numerically in Section 4.

Notice that the boundary scheme we have described above is just for the time level t_n. For the third-order TVD RK method (2.3), we need to go through two intermediate stages $t^{(1)}$ and $t^{(2)}$. In particular, boundary conditions at the intermediate stages are necessary. In [3], the authors pointed out that the boundary conditions should be imposed as follows to maintain the third order accuracy for the fixed boundary case, i.e.,

$a(t) = const,$

$$\begin{aligned} u^n &\sim u(a, t_n), \\ u^{(1)} &\sim u(a, t_n) + \Delta t u_t(a, t_n), \\ u^{(2)} &\sim u(a, t_n) + \frac{\Delta t}{2} u_t(a, t_n) + \frac{\Delta t^2}{4} u_{tt}(a, t_n). \end{aligned} \tag{2.19}$$

Actually, on the fixed boundary we have that $u = g_l$, $u_t = g_l'$ and $u_{tt} = g_l''$. For the moving boundary, we will follow the idea mentioned in [23] to treat the boundary as "fixed" at $x = a(t_n)$ on each stage of the RK method, and use the ILW procedure on this fixed boundary to obtain the ghost point values. Therefore, we need to construct second and first order approximations of $u_t(a(t_n), t_n)$ and $u_{tt}(a(t_n), t_n)$, respectively. After that, we can obtain the point values on $x = a(t_n)$ at intermediate stages $t^{(1)}$ and $t^{(2)}$ via the equations (2.19).

Given the Dirichlet boundary condition and its time derivative (2.9), we can easily have that

$$u(a(t_n), t) = g_l(t_n), \quad \text{and} \quad u_t(a(t_n), t_n) = g_l'(t_n) - a'(t_n) u_x(a(t_n), t_n). \tag{2.20}$$

Here, $u_x(a(t_n), t_n)$ can be replaced by $u^{*(1)}$ obtained via the combination (2.17). To obtain $u_{tt}(a(t_n), t_n)$, we need to take second order derivative to $g_l(t)$ and have ,

$$\begin{aligned} u_{tt}(a(t_n), t_n) =& g_l''(t_n) - a''(t_n) \cdot u_x(a(t_n), t_n) - (a'(t_n))^2 \cdot u_{xx}(a(t_n), t_n) \\ & - 2a'(t_n) \cdot u_{tx}(a(t_n), t_n), \end{aligned} \tag{2.21}$$

where $u_{xx}(a(t_n), t_n)$ can be replaced by $u^{*(2)}$ obtained via (2.17). However, the mixed derivative $u_{tx}(a(t_n), t_n)$ is not known yet. Following the idea in [23], we can get $u_{tx}(a(t_n), t_n)$ through a standard Lax-Wendroff procedure. Using the PDE repeatedly, we can have that

$$u_{tx} = -f'(u) u_{xx} - f''(u)(u_x)^2 + \epsilon u_{xxx},$$

and all the terms in this equation have been obtained by the ILW procedure or extrapolation at time level t_n. However, the formula would be very complicated, especially for high-dimensional systems. Notice that the time derivative $(u_t)_j$ at time $t = t_n$ and at every interior point is already known in the first stage of the RK method. Hence, u_{tx} can be obtained by the spatial extrapolation on $(u_t)_j$. Because we only require u_{tx} to have first order accuracy, we only need to construct a linear polynomial extrapolation, which is much easier.

Here we summarize our algorithm. Our goal is to impose values of ghost points $\{x_{-2}, x_{-1}\}$ at time level t_n and the two intermediate RK stages.

For time level t_n:

Step 1. Get the point value on the boundary $u^{*(0)}$ based on the given boundary condition (2.8). Make a convex combination of the derivatives to compute $u^{*(1)}$ and $u^{*(2)}$ via (2.17).

Step 2. Use the third order Taylor expansion (2.7) at the left boundary to get the values at the ghost points $\{x_{-1}, x_{-2}\}$.

For the first and second intermediate RK stages:

Step 3. Use (2.20) to get $u(a(t_n, t_n))$ and $u_t(a(t_n, t_n))$. Use a linear polynomial extrapolation to get $u_{tx}(a(t_n, t_n))$, and $u_{tt}(a(t_n, t_n))$ can be obtained via (2.21). Then, modify the boundary conditions at the intermediate stages $t^{(1)}$ and $t^{(2)}$ through equations (2.19)

Step 4. Do Steps 1 and 2 at the intermediate stages $t^{(1)}$ and $t^{(2)}$.

2.2 One-dimensional convection-diffusion system with moving boundaries

We consider the one-dimensional system:

$$\begin{cases} \mathbf{U}_t + \mathbf{A}(\mathbf{U})\mathbf{U}_x = \mathbf{B}(\mathbf{U})\mathbf{U}_{xx} + \mathbf{S}(\mathbf{U}), \quad x \in \Omega(t) = (a(t), b(t)),\ t > 0, \\ \mathbf{U}(a(t), t) = \mathbf{G}_l(t), \qquad t > 0, \\ \mathbf{U}(b(t), t) = \mathbf{G}_r(t), \qquad t > 0, \\ \mathbf{U}(x, 0) = \mathbf{U}_0(x), \qquad x \in \Omega(t) = (a(t), b(t)), \end{cases} \tag{2.22}$$

where, $\mathbf{U} = (U_1, U_2, ..., U_m)^T$, $\mathbf{S} = \mathbf{S}(\mathbf{U})$ is the source term, $\mathbf{A}(\mathbf{U})$ and $\mathbf{B}(\mathbf{U})$ are diagonalizable matrices satisfying

$$\mathbf{A}(\mathbf{U}) = \mathbf{L}^{-1}(\mathbf{U})\mathbf{\Lambda}(\mathbf{U})\mathbf{L}(\mathbf{U}), \quad \mathbf{B}(\mathbf{U}) = \mathbf{C}(\mathbf{U})^{-1}\mathbf{D}(\mathbf{U})\mathbf{C}(\mathbf{U}).$$

Here, $\mathbf{\Lambda}(\mathbf{U}) = diag\{\lambda_i(\mathbf{U})\}_{i=1}^m$, $\mathbf{D}(\mathbf{U}) = diag\{\epsilon_i(\mathbf{U})\}_{i=1}^m$ with $\epsilon_i(\mathbf{U}) \geq 0$, and

$$\mathbf{L}(\mathbf{U}) = \begin{pmatrix} \mathbf{l}_1(\mathbf{U}) \\ \mathbf{l}_2(\mathbf{U}) \\ \cdots \\ \mathbf{l}_m(\mathbf{U}) \end{pmatrix}, \qquad \mathbf{C}(\mathbf{U}) = \begin{pmatrix} \mathbf{c}_1(\mathbf{U}) \\ \mathbf{c}_2(\mathbf{U}) \\ \cdots \\ \mathbf{c}_m(\mathbf{U}) \end{pmatrix}. \tag{2.23}$$

As before, we take the left boundary $x = a(t)$ as an example to describe our boundary treatment. Similar as in the scalar case, to get a third order boundary scheme, we can use a third order Taylor expansion to define the values at ghost points near the left boundary $x = a(t_n)$ as follows:

$$\mathbf{U}_j = \mathbf{U}^{*(0)} + (x_j - a(t_n))\mathbf{U}^{*(1)} + \frac{(x_j - a(t_n))^2}{2}\mathbf{U}^{*(2)}, \qquad j = -2, -1, \tag{2.24}$$

and the spatial derivatives $\mathbf{U}^{*(k)}$ approximate $\partial_x^k \mathbf{U}|_{x=a(t_n),t=t_n}$ with $(3-k)$-th order accuracy, respectively. Here, we describe the construction of $\mathbf{U}^{*(0)}, \mathbf{U}^{*(1)}$ and $\mathbf{U}^{*(2)}$ at time level t_n sequentially.

For $\mathbf{U}^{*(0)}$, we can directly obtain its value from the boundary condition

$$\mathbf{U}^{*(0)} = \mathbf{G}_l(t_n).$$

For $\mathbf{U}^{*(1)}$, we will treat the system as convection-dominated to construct the first derivative via the ILW procedure. At first, we perform a characteristic decomposition on the boundary $x = a(t_n)$, and look at the characteristic variables $\mathbf{V} = \mathbf{L}(\mathbf{U}^{*(0)})\mathbf{U}$. Then the equation (2.22) becomes

$$\mathbf{V}_t + \mathbf{\Lambda}(\mathbf{U}^{*(0)})\mathbf{V}_x = \mathbf{RHS}^V = \mathbf{L}(\mathbf{U}^{*(0)})\left(\mathbf{B}(\mathbf{U}^{*(0)})\mathbf{U}_{xx} + \mathbf{S}(\mathbf{U}^{*(0)})\right). \tag{2.25}$$

Taking the time derivative on the boundary condition, we have

$$\mathbf{V}_t|_{x=a(t_n),t=t_n} = \mathbf{L}(\mathbf{U}^{*(0)})\mathbf{G}_l'(t_n) - a'(t_n)\mathbf{V}_x.$$

Combining these two equations, we can obtain that

$$\mathbf{L}(\mathbf{U}^{*(0)})\mathbf{G}_l'(t_n) + (\mathbf{\Lambda}(\mathbf{U}^{*(0)}) - a'(t_n)\mathbf{I})\mathbf{V}_x = \mathbf{RHS}^V, \tag{2.26}$$

where $\mathbf{I}$ is the identity matrix. We will look at each component of $\mathbf{V}$,

$$\mathbf{l}_i(\mathbf{U}^{*(0)}) \cdot \mathbf{G}_l'(t_n) + (\lambda_i(\mathbf{U}^{*(0)}) - a'(t_n))(V_i)_x = RHS_i^V \quad i = 1, ..., m. \tag{2.27}$$

Then, we apply the scalar boundary treatment to construct the corresponding point value and spatial derivatives at $x = a(t_n)$. As before, a convex combination of the terms $(V_i)_{ilw}^{*(1)}$ and $(V_i)_{ext}^{*(1)}$ is designed to ensure the stability of our boundary treatment:

$$(V_i)^{*(1)} = (\omega_i)_{ilw}^{(1)} \cdot (V_i)_{ilw}^{*(1)} + (\omega_i)_{ext}^{(1)} \cdot (V_i)_{ext}^{*(1)}, \quad i = 1, ..., m. \tag{2.28}$$

In order to offset the possible zero in the denominator, in which case the velocity of the moving boundary is close to the velocity of the characteristic variable near the boundary and the diffusion is very weak at the boundary, we set the weights of the combination as

$$\begin{aligned} (\omega_i)_{ilw}^{(1)} &= \frac{\left(\lambda_i(\mathbf{U}^{*(0)}) - a'(t_n)\right)^2 \Delta x^2}{\left(\lambda_i(\mathbf{U}^{*(0)}) - a'(t)\right)^2 \Delta x^2 + (3\epsilon + \mu)^2}, \\ (\omega_i)_{ext}^{(1)} &= 1 - (\omega_i)_{ilw}^{(1)} = \frac{(3\epsilon + \mu)^2}{\left(\lambda_i(\mathbf{U}^{*(0)}) - a'(t)\right)^2 \Delta x^2 + (3\epsilon + \mu)^2}, \end{aligned} \tag{2.29}$$

where $\epsilon = \max_i\{\epsilon_i(\mathbf{U}^{*(0)})\}$ and $\mu = 10^{-6}$. To construct $(V_i)_{ilw}^{*(1)}$, as before, for the incoming variable, we use equation (2.27) to get its first derivative, and for outgoing variable, an extrapolation is used. More specifically, we set

$$(\omega_i)_{ilw}^{(1)} \cdot (V_i)_{ilw}^{*(1)} = (\omega_i)_{ilw}^{(1)} \cdot (V_i)_{ext}^{*(1)} \tag{2.30a}$$

for $\lambda_i(\mathbf{U}^{*(0)}) - a'(t_n) \leq 0$, and

$$(\omega_i)_{ilw}^{(1)} \cdot (V_i)_{ilw}^{*(1)} = \frac{\left(\lambda_i(\mathbf{U}^{*(0)}) - a'(t_n)\right) \Delta x^2}{\left(\lambda_i(\mathbf{U}^{*(0)}) - a'(t_n)\right)^2 \Delta x^2 + (3\epsilon + \mu)^2} \left(-\mathbf{l}_i(\mathbf{U}^{*(0)}) \cdot \mathbf{G}_l'(t_n) + RHS_i^V\right), \tag{2.30b}$$

for $\lambda_i(\mathbf{U}^{*(0)}) - a'(t_n) > 0$. Once we get $\mathbf{V}^{*(1)}$, we transform back into physical space by

$$\mathbf{U}^{*(1)} = \mathbf{L}^{-1}(\mathbf{U}^{*(0)})\mathbf{V}^{*(1)}.$$

As for $\mathbf{U}^{*(2)}$, let $\mathbf{W} = \mathbf{C}(\mathbf{U}^{*(0)})\mathbf{U}$ and construct the second derivative via the ILW procedure. The original system (2.22) can be rewritten as

$$\mathbf{W}_t - \mathbf{D}(\mathbf{U}^{*(0)})\mathbf{W}_{xx} = \mathbf{RHS}^W = \mathbf{C}(\mathbf{U}^{*(0)})[\mathbf{S}(\mathbf{U}^{*(0)}) - \mathbf{A}(\mathbf{U}^{*(0)})\mathbf{U}_x] \tag{2.31}$$

near the boundary. Taking the time derivative on the boundary condition, we have

$$\mathbf{W}_t = \mathbf{C}(\mathbf{U}^{*(0)})\mathbf{G}_l'(t_n) - a'(t_n)\mathbf{W}_x \tag{2.32}$$

Combining (2.31) and (2.32), we can obtain that

$$\mathbf{D}(\mathbf{U}^{*(0)})\mathbf{W}_{xx} = \mathbf{C}(\mathbf{U}^{*(0)})\mathbf{G}_l'(t_n) - a'(t_n)\mathbf{W}_x - \mathbf{RHS}^W, \tag{2.33}$$

which will help us to construct $(W_i)_{ilw}^{*(2)}$, $i = 1, \ldots, m$. Again, we give a convex combination of $(W_i)_{ilw}^{*(2)}$ and $(W_i)_{ext}^{*(2)}$,

$$(W_i)^{*(2)} = (\omega_i)_{ilw}^{(2)} \cdot (W_i)_{ilw}^{*(2)} + (\omega_i)_{ext}^{(2)} \cdot (W_i)_{ext}^{*(2)}, \quad i = 1, \ldots, m, \tag{2.34}$$

with the weights

$$\begin{aligned} (\omega_i)_{ilw}^{(2)} &= \frac{9\epsilon_i^2(\mathbf{U}^{*(0)})}{(\alpha+\mu)^2\Delta x^2 + 9\epsilon_i^2(\mathbf{U}^{*(0)})}, \\ (\omega_i)_{ext}^{(2)} &= 1 - (\omega_i)_{ilw}^{(2)} = \frac{(\alpha+\mu)^2\Delta x^2}{(\alpha+\mu)^2\Delta x^2 + 9\epsilon_i^2(\mathbf{U}^{*(0)})}. \end{aligned} \tag{2.35}$$

Here, $\alpha = \max_i\{|\lambda_i(\mathbf{U}^{*(0)}) - a'(t_n)|\}$. The previous technique is applied when we compute $(\omega_i)_{ilw}^{(2)} \cdot (W_i)_{ilw}^{*(2)}$, which can be written as:

$$(\omega_i)_{ilw}^{(2)} \cdot (W_i)_{ilw}^{*(2)} = \frac{9\epsilon_i(\mathbf{U}^{*(0)})}{(\alpha+\mu)^2\Delta x^2 + 9\epsilon_i(\mathbf{U}^{*(0)})^2} \left(\mathbf{c}_i(\mathbf{U}^{*(0)}) \cdot \mathbf{G}_l'(t_n) - a'(t_n)(W_i)_{ext}^{*(1)} - RHS_i^W\right). \tag{2.36}$$

Finally, we transform back and obtain

$$\mathbf{U}^{*(2)} = \mathbf{C}^{-1}(\mathbf{U}^{*(0)})\mathbf{W}^{*(2)}. \tag{2.37}$$

2.3 Two-dimensional convection-diffusion system with moving boundaries

We now consider the two-dimensional convection-diffusion system

$$\begin{cases} \mathbf{U}_t + \mathbf{A}_1\mathbf{U}_x + \mathbf{A}_2\mathbf{U}_y = \mathbf{B}_1\mathbf{U}_{xx} + \mathbf{B}_2\mathbf{U}_{xy} + \mathbf{B}_3\mathbf{U}_{yy} + \mathbf{S}, \quad (x,y) \in \Omega(t),\, t > 0 \\ \mathbf{U}(x,y,t) = \mathbf{G}(x,y,t), \quad (x,y) \in \partial\Omega(t),\, t > 0, \\ \mathbf{U}(x,y,0) = \mathbf{U}_0(x,y), \quad (x,y) \in \Omega(t), \end{cases} \tag{2.38}$$

where, $\mathbf{U} = (U_1, U_2, ..., U_m)^T$, $\mathbf{S} = \mathbf{S}(\mathbf{U})$ is the source term, $\mathbf{A_1} = \mathbf{A_1}(\mathbf{U}), \mathbf{A_2} = \mathbf{A_2}(\mathbf{U}), \mathbf{B_1} = \mathbf{B_1}(\mathbf{U}), \mathbf{B_2} = \mathbf{B_2}(\mathbf{U})$ and $\mathbf{B_3} = \mathbf{B_3}(\mathbf{U})$ are matrices related to $\mathbf{U}$. In particular, when $m = 1$, it is a scalar equation. We assume the convection term $\mathbf{A}_1\mathbf{U}_x + \mathbf{A}_2\mathbf{U}_y$ makes the system $\mathbf{U}_t + \mathbf{A}_1\mathbf{U}_x + \mathbf{A}_2\mathbf{U}_y = 0$ to be strongly hyperbolic and the diffusion term $\mathbf{B}_1\mathbf{U}_{xx} + \mathbf{B}_2\mathbf{U}_{xy} + \mathbf{B}_3\mathbf{U}_{yy}$ makes the system $\mathbf{U}_t = \mathbf{B}_1\mathbf{U}_{xx} + \mathbf{B}_2\mathbf{U}_{xy} + \mathbf{B}_3\mathbf{U}_{yy}$ to be parabolic.

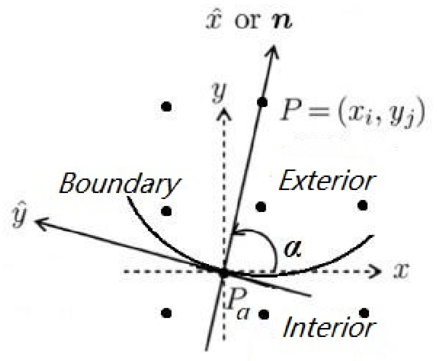


Figure 1: The local coordinate rotation diagram.

As before, we use a high order finite difference method to discrete spatial derivatives of (2.38) on a fixed Cartesian mesh (x_i, y_j) with uniform mesh sizes Δx and Δy in each direction for all time. Ghost points may be needed near the boundary $\partial\Omega(t)$. Hence, we need to design a method to define their values.

At time level t_n, assume $P_{ij} = (x_i, y_j)$ is a ghost point near the boundary. At first, we find its pedal $P_a \in \partial\Omega(t_n)$, so that the normal $\mathbf{n}$ at P_a goes through P_{ij}, as shown in Figure 1. Assume the normal vector $\mathbf{n} = (\cos\alpha, \sin\alpha)^T$. In order to simplify the algorithm, we perform the following local coordinate rotation transformation at P_a by

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{2.39}$$

In this new coordinate system, the system (2.38) can be rewritten as follows:

$$\mathbf{U}_t + \hat{\mathbf{A}}_1(\mathbf{U})\mathbf{U}_{\hat{x}} + \hat{\mathbf{A}}_2(\mathbf{U})\mathbf{U}_{\hat{y}} = \hat{\mathbf{B}}_1(\mathbf{U})\mathbf{U}_{\hat{x}\hat{x}} + \hat{\mathbf{B}}_2(\mathbf{U})\mathbf{U}_{\hat{x}\hat{y}} + \hat{\mathbf{B}}_3(\mathbf{U})\mathbf{U}_{\hat{y}\hat{y}} + \mathbf{S} \tag{2.40}$$

where

$$\begin{aligned}
\hat{\mathbf{A}}_1 &= \cos\alpha\,\mathbf{A}_1 + \sin\alpha\,\mathbf{A}_2,\\
\hat{\mathbf{A}}_2 &= -\sin\alpha\,\mathbf{A}_1 + \cos\alpha\,\mathbf{A}_2,\\
\hat{\mathbf{B}}_1 &= \cos^2\alpha\,\mathbf{B}_1 + \cos\alpha\,\sin\alpha\,\mathbf{B}_2 + \sin^2\alpha\,\mathbf{B}_3,\\
\hat{\mathbf{B}}_2 &= -2\cos\alpha\,\sin\alpha\,\mathbf{B}_1 + (\cos^2\alpha - \sin^2\alpha)\,\mathbf{B}_2 + 2\cos\alpha\,\sin\alpha\,\mathbf{B}_3,\\
\hat{\mathbf{B}}_3 &= \sin^2\alpha\,\mathbf{B}_1 - \cos\alpha\,\sin\alpha\,\mathbf{B}_2 + \cos^2\alpha\,\mathbf{B}_3.
\end{aligned}$$

Define

$$\mathbf{Res} = -\hat{\mathbf{A}}_2(\mathbf{U})\mathbf{U}_{\hat{y}} + \hat{\mathbf{B}}_2(\mathbf{U})\mathbf{U}_{\hat{x}\hat{y}} + \hat{\mathbf{B}}_3(\mathbf{U})\mathbf{U}_{\hat{y}\hat{y}} + \mathbf{S}.$$

Then, we can write (2.40) as

$$\mathbf{U}_t + \mathbf{A}(\mathbf{U})\mathbf{U}_{\hat{x}} = \mathbf{B}(\mathbf{U})\mathbf{U}_{\hat{x}\hat{x}} + \mathbf{Res}, \tag{2.41}$$

where $\mathbf{A} = \hat{\mathbf{A}}_1$, $\mathbf{B} = \hat{\mathbf{B}}_1$ are diagonalizable matrices satisfying

$$\mathbf{A}(\mathbf{U}) = \mathbf{L}(\mathbf{U})^{-1}\mathbf{\Lambda}(\mathbf{U})\mathbf{L}(\mathbf{U}), \quad \mathbf{B}(\mathbf{U}) = \mathbf{C}(\mathbf{U})^{-1}\mathbf{D}(\mathbf{U})\mathbf{C}(\mathbf{U}). \tag{2.42}$$

Here, $\mathbf{\Lambda}(\mathbf{U}) = diag\{\lambda_i(\mathbf{U})\}$, $\mathbf{D}(\mathbf{U}) = diag\{\epsilon_i(\mathbf{U})\}$ with $\epsilon_i(\mathbf{U}) \geq 0$, and

$$\mathbf{L}(\mathbf{U}) = \begin{pmatrix} \mathbf{l}_1(\mathbf{U}) \\ \mathbf{l}_2(\mathbf{U}) \\ \cdots \\ \mathbf{l}_m(\mathbf{U}) \end{pmatrix}, \qquad \mathbf{C}(\mathbf{U}) = \begin{pmatrix} \mathbf{c}_1(\mathbf{U}) \\ \mathbf{c}_2(\mathbf{U}) \\ \cdots \\ \mathbf{c}_m(\mathbf{U}) \end{pmatrix}. \tag{2.43}$$

For a third order boundary treatment, the value of the ghost point $P_{i,j}$ is imposed by the Taylor expansion in the $\hat{x}$-direction

$$\mathbf{U}_{i,j} = \mathbf{U}^{*(0)} + |P_{i,j} - P_a|\mathbf{U}^{*(1)} + \frac{1}{2}|P_{i,j} - P_a|^2\mathbf{U}^{*(2)}, \tag{2.44}$$

where $\mathbf{U}^{*(k)}$ is an approximation of the normal derivative $\partial_{\hat{x}}^k\mathbf{U}|_{(x,y)=P_a,t=t_n}$ with $(3-k)$-th order of accuracy, $k = 0, 1, 2$.

To describe the boundary conditions, let $\mathbf{X}_b = \mathbf{X}_b(s, t)$ present the position vector (in Eulerian coordinates) of a point on the moving boundary $\partial\Omega(t)$, where s is the Lagrangian coordinate of the boundary, and t is the current moment. The velocity of the boundary can be written as $\mathbf{V}_b = \frac{\partial \mathbf{X}_b}{\partial t} = (u_b, v_b)^T$. Assuming that $P_a = \mathbf{X}_b(s_a, t_n)$. The boundary condition at P_a can be written as $\widetilde{\mathbf{G}}(t) = \mathbf{G}(\mathbf{X}_b(s_d, t), t)$.

Now, we describe the construction of $\mathbf{U}^{*(0)}, \mathbf{U}^{*(1)}, \mathbf{U}^{*(2)}$ at time level t_n sequentially, following the idea of the one-dimensional case.

For $\mathbf{U}^{*(0)}$, we can directly obtain its value from the boundary conditions

$$\mathbf{U}^{*(0)} = \widetilde{\mathbf{G}}(t_n).$$

For $\mathbf{U}^{*(1)}$, we perform a characteristic decomposition firstly, i.e., let $\mathbf{V} = \mathbf{L}(\mathbf{U}^{*(0)})\mathbf{U}$, where $\mathbf{V}$ is the characteristic variable. Then, we have the equation

$$\mathbf{V}_t + \mathbf{\Lambda}(\mathbf{U}^{*(0)})\mathbf{V}_{\hat{x}} = \mathbf{RHS}^V = \mathbf{L}(\mathbf{U}^{*(0)})(\mathbf{B}\mathbf{U}_{\hat{x}\hat{x}} + \mathbf{Res}). \tag{2.45}$$

At the boundary P_a, we have

$$\mathbf{L}(\mathbf{U}^{*(0)})\widetilde{\mathbf{G}}'(t_n) = \frac{D}{Dt}\mathbf{V} = \mathbf{V}_t + \mathbf{V}_{\hat{x}}\hat{u}_b + \mathbf{V}_{\hat{y}}\hat{v}_b,$$

with

$$\begin{pmatrix} \hat{u}_b \\ \hat{v}_b \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} u_b \\ v_b \end{pmatrix}.$$

Therefore, we have

$$(\mathbf{\Lambda}(\mathbf{U}^{*(0)}) - \hat{u}_b\mathbf{I})\mathbf{V}_x = -\mathbf{L}(\mathbf{U}^{*(0)})\widetilde{\mathbf{G}}'(t_n) + \mathbf{RHS}^V + \mathbf{V}_{\hat{y}}\hat{v}_b. \tag{2.46}$$

Similar to the one-dimensional case, we set:

$$(V_i)^{*(1)} = (\omega_i)^{(1)}_{ilw} \cdot (V_i)^{*(1)}_{ilw} + (\omega_i)^{(1)}_{ext} \cdot (V_i)^{*(1)}_{ext}, \quad i = 1, \ldots, m, \tag{2.47}$$

where

$$(\omega_i)^{(1)}_{ilw} = \frac{(\lambda_i(\mathbf{U}^{*(0)}) - \hat{u}_b)^2 h^2}{(\lambda_i(\mathbf{U}^{*(0)}) - \hat{u}_b)^2 h^2 + (3\epsilon + \mu)^2},$$
$$(\omega_i)^{(1)}_{ext} = 1 - (\omega_i)^{(1)}_{ilw} = \frac{(3\epsilon + \mu)^2}{(\lambda_i(\mathbf{U}^{*(0)}) - \hat{u}_b)^2 h^2 + (3\epsilon + \mu)^2}, \tag{2.48}$$

with $\epsilon_i = \max\{|\epsilon_i(\mathbf{U}^{*(0)})|\}$, $\mu = 10^{-6}$ and $h = \sqrt{\Delta x^2 + \Delta y^2}$. In particular, we set

$$(\omega_i)^{(1)}_{ilw} \cdot (V_i)^{*(1)}_{ilw} = (\omega_i)^{(1)}_{ilw} \cdot (V_i)^{*(1)}_{ext} \tag{2.49a}$$

for $\lambda_i(\mathbf{U}^{*(0)}) - \hat{u}_b \geq 0$, and

$$(\omega_i)^{(1)}_{ilw} \cdot (V_i)^{*(1)}_{ilw} = \frac{(\lambda_i(\mathbf{U}^{*(0)}) - \hat{u}_b) h^2}{(\lambda_i(\mathbf{U}^{*(0)}) - \hat{u}_b)^2 h^2 + (3\epsilon + \mu)^2} \left(-\mathbf{l}_i(\mathbf{U}^{*(0)}) \cdot \widetilde{\mathbf{G}}'(t_n) + RHS^V_i + (V_i)_{\hat{y}} \hat{v}_b \right), \tag{2.49b}$$

for $\lambda_i(\mathbf{U}^{*(0)}) - \hat{u}_b < 0$. All the derivatives in RHS^V_i and $(V_i)_{\hat{y}}$ are constructed by extrapolation. Finally,

$$\mathbf{U}^{*(1)} = \mathbf{L}^{-1}(\mathbf{U}^{*(0)}) \mathbf{V}^{*(1)}.$$

As for $\mathbf{U}^{*(2)}$, let $\mathbf{W} = \mathbf{C}\mathbf{U}$, then we have

$$\mathbf{W}_t - \mathbf{D}(\mathbf{U}^{*(0)}) \mathbf{W}_{\hat{x}\hat{x}} = \mathbf{RHS}^W = \mathbf{C}(\mathbf{U}^{*(0)})[\mathbf{Res} - \mathbf{A}(\mathbf{U}^{*(0)}) \mathbf{U}_{\hat{x}}]. \tag{2.50}$$

Since

$$\mathbf{W}_t = \mathbf{C}(\mathbf{U}^{*(0)}) \mathbf{G}'(t_n) - \mathbf{W}_{\hat{x}} \hat{u}_b - \mathbf{W}_{\hat{y}} \hat{v}_b,$$

we have

$$\mathbf{D}(\mathbf{U}^{*(0)}) \mathbf{W}_{xx} = \mathbf{C}(\mathbf{U}^{*(0)}) \mathbf{G}'(t_n) - \mathbf{W}_{\hat{x}} \hat{u}_b - \mathbf{W}_{\hat{y}} \hat{v}_b - \mathbf{RHS}^W. \tag{2.51}$$

Therefore, we give the convex combination

$$(W_i)^{*(2)} = (\omega_i)^{(2)}_{ilw} \cdot (W_i)^{*(2)}_{ilw} + (\omega_i)^{(2)}_{ext} \cdot (W_i)^{*(2)}_{ext}, \tag{2.52}$$

where

$$(\omega_i)^{(2)}_{ilw} \cdot (W_i)^{*(2)}_{ilw} = \frac{9\epsilon_i^2(\mathbf{U}^{*(0)})}{(\alpha + \mu)^2 h^2 + 9\epsilon_i(\mathbf{U}^{*(0)})^2} \left(\mathbf{c}_i(\mathbf{U}^{*(0)}) \cdot \mathbf{G}'(t_n) - (W_i)_{\hat{x}} \hat{u}_b - (W_i)_{\hat{y}} \hat{v}_b - RHS^W_i \right), \tag{2.53}$$

and

$$(\omega_i)_{ext}^{(2)} = \frac{(\alpha+\mu)^2h^2}{(\alpha+\mu)^2h^2 + 9\epsilon_i^2(\mathbf{U}^{*(0)})}. \tag{2.54}$$

Here, $\alpha = \max_i\{|\lambda_i(\mathbf{U}^{*(0)}) - \hat{u}_b|\}$, $\mu = 10^{-6}$ and $h = \sqrt{\Delta x^2 + \Delta y^2}$. Finally,

$$\mathbf{U}^{*(2)} = \mathbf{C}^{-1}(\mathbf{U}^{*(0)})\mathbf{W}^{*(2)}.$$

As before, when we use the third order TVD Runge-Kutta to update our solution, in order to maintain the third order accuracy, we should modify the boundary condition at the two intermediate stages $t^{(1)}$ and $t^{(2)}$ via (2.19). So we should obtain the values of $\mathbf{U}_t|_{t=t_n}$, $\mathbf{U}_{tt}|_{t=t_n}$ at the boundary P_a.

As we know

$$\widetilde{\mathbf{G}}'(t_n) = (\mathbf{U}_t + \mathbf{U}_{\hat{x}}\hat{u}_b + \mathbf{U}_{\hat{y}}\hat{v}_b)|_{(x,y)=P_a,t=t_n}. \tag{2.55}$$

Replace $\mathbf{U}_{\hat{x}}$ by $\mathbf{U}^{*(1)}$, then we have an approximation of $\mathbf{U}_t|_{(x,y)=P_a,t=t_n}$ as

$$\mathbf{U}_t|_{(x,y)=P_a,t=t_n} = \mathbf{G}'(t_n) - \mathbf{U}^{*(1)}\hat{u}_b - \mathbf{U}_{\hat{y}}|_{(x,y)=P_a,t=t_n}\hat{v}_b. \tag{2.56}$$

Also,

$$\begin{aligned}\widetilde{\mathbf{G}}''(t_n) = \frac{\partial \hat{u}_b}{\partial t}\mathbf{U}_{\hat{x}} + \frac{\partial \hat{v}_b}{\partial t}\mathbf{U}_{\hat{y}} + \hat{u}_b^2\mathbf{U}_{\hat{x}\hat{x}} + 2\hat{u}_b\hat{v}_b\mathbf{U}_{\hat{x}\hat{y}} + \hat{v}_b^2\mathbf{U}_{\hat{y}\hat{y}} + \\ 2(\hat{u}_b\mathbf{U}_{t\hat{x}} + \hat{v}_b\mathbf{U}_{t\hat{y}}) + \mathbf{U}_{tt},\end{aligned} \tag{2.57}$$

where $(\frac{\partial \hat{u}_b}{\partial t}, \frac{\partial \hat{v}_b}{\partial t})$ is the acceleration of the moving boundary. So, we have

$$\begin{aligned}\mathbf{U}_{tt}|_{(x,y)=P_a,t=t_n} = \widetilde{\mathbf{G}}''(t_n) - \Bigg(\frac{\partial \hat{u}_b}{\partial t}\mathbf{U}_{\hat{x}} + \frac{\partial \hat{v}_b}{\partial t}\mathbf{U}_{\hat{y}} + \hat{u}_b^2\mathbf{U}_{\hat{x}\hat{x}} + 2\hat{u}_b\hat{v}_b\mathbf{U}_{\hat{x}\hat{y}} + \hat{v}_b^2\mathbf{U}_{\hat{y}\hat{y}} + \\ 2(\hat{u}_b\mathbf{U}_{t\hat{x}} + \hat{v}_b\mathbf{U}_{t\hat{y}})\Bigg),\end{aligned} \tag{2.58}$$

where $\mathbf{U}_{\hat{x}}$, $\mathbf{U}_{\hat{y}}$, $\mathbf{U}_{\hat{x}\hat{x}}$, $\mathbf{U}_{\hat{x}\hat{y}}$, and $\mathbf{U}_{\hat{y}\hat{y}}$ have been obtained at time level t_n. Therefore, we only need to deal with two mixed derivative terms $\mathbf{U}_{t\hat{x}}$ and $\mathbf{U}_{t\hat{y}}$. Note that the value of $(\mathbf{U}_t)_{i,j}$ is already known at every grid point inside the computational domain in the first stage of RK, so $\mathbf{U}_{t\hat{x}}$ and $\mathbf{U}_{t\hat{y}}$ can be obtained by linear extrapolation, because we only require first-order accuracy of $\mathbf{U}_{tt}|_{(x,y)=P_a,t=t_n}$ to obtain the overall third-order accuracy for the scheme.

3 Application to moving rigid body in viscous flows

3.1 Problem description

We consider the two-dimensional *Navier-Stokes (NS) equation*,

$$\mathbf{W}_t + \mathbf{F}(\mathbf{W})_x + \mathbf{G}(\mathbf{W})_y = \frac{1}{Re}\left(\mathbf{S_1}(\mathbf{W})_x + \mathbf{S_2}(\mathbf{W})_y\right), \quad \mathbf{x} = (x, y) \in \Omega(t), \tag{3.1}$$

where

$$\mathbf{W} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \quad \mathbf{F}(\mathbf{W}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}, \quad \mathbf{G}(\mathbf{W}) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}. \tag{3.2}$$

Here, ρ, $\mathbf{u} = (u, v)^T$, p and E represent the density, velocity, pressure and total energy per volume, respectively. An equation of state relates the pressure and the other variables is given

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho(u^2 + v^2)\right). \tag{3.3}$$

Here, γ is the adiabatic constant, which equals to 1.4 for an ideal polytropic gas.

The terms on the right hand side of (3.1) are in the form of

$$\mathbf{S_1}(\mathbf{W}) = \begin{pmatrix} 0 \\ \tau_{11} \\ \tau_{21} \\ \sigma_1 \end{pmatrix}, \quad \mathbf{S_2}(\mathbf{W}) = \begin{pmatrix} 0 \\ \tau_{12} \\ \tau_{22} \\ \sigma_2 \end{pmatrix},$$

with the components of the viscous stress tensor given by

$$\begin{aligned} \tau_{11} &= \frac{4}{3}u_x - \frac{2}{3}v_y, \\ \tau_{22} &= \frac{4}{3}v_y - \frac{2}{3}u_x, \\ \tau_{12} &= \tau_{21} = v_x + u_y, \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} \sigma_1 &= u\,\tau_{11} + v\,\tau_{12} + \frac{(c^2)_x}{(\gamma - 1)Pr}, \\ \sigma_2 &= u\,\tau_{21} + v\,\tau_{22} + \frac{(c^2)_y}{(\gamma - 1)Pr}. \end{aligned} \tag{3.5}$$

Here, Pr is the Prandtl number, Re is the Reynolds number, $T = p/\rho$ is the temperature, and $c = \sqrt{\gamma p/\rho}$ is the sound speed.

We consider a moving rigid body in viscous flows, with the *isothermal no-slip wall boundary condition* on the boundary of the rigid body $\Gamma(t) \subset \partial\Omega(t)$, which can be written as:

$$\begin{cases} \mathbf{u} = \mathbf{V}_b, \\ T = T_b, \end{cases} \tag{3.6}$$

where, $\mathbf{V}_b = (u_b, v_b)^T$ is the boundary velocity, T_b is the temperature of the boundary, and $\mathbf{n}$ is normal vector point to the rigid body from the fluid.

Suppose the motion of a rigid body is induced by the viscous flow. Both translation and rotation of the body should be taken into account. For each point $\mathbf{X}_b = (x_b, y_b) \in \Gamma(t)$, the velocity is given as

$$\mathbf{V}_b = \mathbf{V}_{tr} + \omega \times \mathbf{r},$$

where, $\mathbf{V}_{tr}$ is the translational velocity, ω is the rotation velocity, and $\mathbf{r}$ is the vector from the boundary point to the centroid of the rigid body. Since the motion of a rigid body is induced by the fluid, $\mathbf{V}_{tr}$ and ω satisfy the equations

$$\frac{d\mathbf{X}_b}{dt} = \mathbf{V}_b, \quad \frac{d\mathbf{V}_{tr}}{dt} = \mathbf{a}_{tr}, \quad \frac{d\omega}{dt} = \mathbf{a}_\theta, \tag{3.7}$$

with the translational acceleration $\mathbf{a}_{tr}$ and rotational acceleration $\mathbf{a}_\theta$ determined by Newton's second law and rotational law respectively.

$$\begin{aligned} M\mathbf{a}_{tr} &= \oint_{\Gamma(t)} (p\,\mathbf{n} - \boldsymbol{\tau}\cdot\mathbf{n})\,dS, \\ I\mathbf{a}_\theta &= \oint_{\Gamma(t)} \mathbf{r} \times (p\,\mathbf{n} - \boldsymbol{\tau}\cdot\mathbf{n})\,dS. \end{aligned} \tag{3.8}$$

Here, M is the rigid body mass and I is the moment. The matrix $\boldsymbol{\tau}$ is defined as $\boldsymbol{\tau} = \frac{1}{Re}[\tau_{i,j}]_{2\times 2}$.

3.2 Numerical schemes

Again, we assume the domain $\Omega(t)$ is covered by a fixed Cartesian mesh with uniform mesh sizes Δx and Δy in each direction. The semi-discrete scheme of the NS equations (3.1) is given as

$$\frac{d\mathbf{W}_{i,j}(t)}{dt} + \frac{\hat{\mathbf{F}}_{i+1/2,j} - \hat{\mathbf{F}}_{i-1/2,j}}{\Delta x} + \frac{\hat{\mathbf{G}}_{i,j+1/2} - \hat{\mathbf{G}}_{i,j-1/2}}{\Delta y} = \mathbf{S}_{i,j}. \tag{3.9}$$

where $\hat{\mathbf{F}}_{i+1/2,j}$ and $\hat{\mathbf{G}}_{i,j+1/2}$ are numerical fluxes obtained by the third order finite difference WENO scheme. For the right hand term $\mathbf{S}_{i,j}$, we use fourth order central finite difference scheme. Specifically, the three different types of derivatives are given below as examples:

$$\begin{aligned}
\frac{\partial f}{\partial x}|_{i,j} =& \frac{1}{12\Delta x}(f_{i-2,j} - 8f_{i-1,j} + 8f_{i+1,j} - f_{i+2,j}),\\
\frac{\partial^2 f}{\partial x^2}|_{i,j} =& \frac{1}{12\Delta x^2}(-f_{i-2,j} + 16f_{i-1,j} - 30f_{i,j} + 16f_{i+1,j} - f_{i+2,j}),\\
\frac{\partial^2 f}{\partial x\partial y}|_{i,j} =& \frac{1}{144\Delta x\Delta y}((f_{i-2,j-2} - 8f_{i-1,j-2} + 8f_{i+1,j-2} - f_{i+2,j-2})\\
& - 8(f_{i-2,j-1} - 8f_{i-1,j-1} + 8f_{i+1,j-1} - f_{i+2,j-1})\\
& + 8(f_{i-2,j+1} - 8f_{i-1,j+1} + 8f_{i+1,j+1} - f_{i+2,j+1})\\
& - (f_{i-2,j+2} - 8f_{i-1,j+2} + 8f_{i+1,j+2} - f_{i+2,j+2})).
\end{aligned} \tag{3.10}$$

Again, we use the TVD Runge-Kutta method (2.3) for the time discretization.

At time level t_n, to make the interior finite difference scheme work, we need to define values of ghost points near the boundary. As before, assume P_{ij} is a ghost point near the boundary and P_a is its pedal on the boundary. With the same rotation as in (2.39), let

$$\begin{pmatrix}\hat{x}\\ \hat{y}\end{pmatrix} = \begin{pmatrix}\cos\alpha & \sin\alpha\\ -\sin\alpha & \cos\alpha\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}, \quad \hat{\mathbf{u}} = \begin{pmatrix}\hat{u}\\ \hat{v}\end{pmatrix} = \begin{pmatrix}\cos\alpha & \sin\alpha\\ -\sin\alpha & \cos\alpha\end{pmatrix}\begin{pmatrix}u\\ v\end{pmatrix}.$$

The Dirichlet boundary conditions are given for the velocity $\hat{\mathbf{u}}$ and the temperature T, so we rewrite the equation (3.1) with respect to the new variables:

$$\hat{\mathbf{U}} = (\rho, \hat{u}, \hat{v}, T)^T,$$

and then we get following system

$$\hat{\mathbf{U}}_t + \mathbf{A}(\hat{\mathbf{U}})\hat{\mathbf{U}}_{\hat{x}} = \mathbf{B}(\hat{\mathbf{U}})\hat{\mathbf{U}}_{\hat{x}\hat{x}} + \mathbf{Res}, \tag{3.11}$$

where

$$\mathbf{A}(\hat{\mathbf{U}}) = \begin{pmatrix}\hat{u} & \rho & 0 & 0\\ \frac{T}{\rho} & \hat{u} & 0 & 1\\ 0 & 0 & \hat{u} & 0\\ 0 & (\gamma-1)T & 0 & \hat{u}\end{pmatrix}, \quad \mathbf{B}(\hat{\mathbf{U}}) = \begin{pmatrix}0 & 0 & 0 & 0\\ 0 & \frac{4}{3Re\cdot\rho} & 0 & 0\\ 0 & 0 & \frac{1}{Re\cdot\rho} & 0\\ 0 & 0 & 0 & \frac{\gamma}{Pr\cdot Re\cdot\rho}\end{pmatrix}, \tag{3.12}$$

and

$$\mathbf{Res} = \begin{pmatrix} Res_1 \\ Res_2 \\ Res_3 \\ Res_4 \end{pmatrix} = \begin{pmatrix} -\rho\hat{v}_{\hat{y}} - \hat{v}\rho_{\hat{y}} \\ \frac{1}{Re\cdot\rho}(\hat{u}_{\hat{y}\hat{y}} + \frac{1}{3}\hat{v}_{\hat{x}\hat{y}} - \hat{v}\hat{u}_{\hat{y}}) \\ \frac{1}{Re\cdot\rho}(\frac{4}{3}\hat{v}_{\hat{y}\hat{y}} + \frac{1}{3}\hat{u}_{\hat{x}\hat{y}} - \frac{1}{\rho}P_{\hat{y}} - \hat{v}\hat{v}_{\hat{y}}) \\ \frac{\gamma}{Re\cdot\rho\cdot Pr}T_{\hat{y}\hat{y}} + NLT \end{pmatrix}, \tag{3.13}$$

with

$$NLT = \frac{\gamma - 1}{Re\cdot\rho}\left(-\frac{2}{3}(\hat{u}_{\hat{x}} + \hat{v}_{\hat{y}} + 2(\hat{u}_{\hat{x}}^2 + \hat{v}_{\hat{y}}^2) + 2(\hat{v}_{\hat{x}} + \hat{u}_{\hat{y}})^2)\right).$$

Here, the matrix $\mathbf{A}(\hat{\mathbf{U}})$ is diagonalizable,

$$\mathbf{A}(\hat{\mathbf{U}}) = \mathbf{L}^{-1}(\hat{\mathbf{U}})\mathbf{\Lambda}(\hat{\mathbf{U}})\mathbf{L}(\hat{\mathbf{U}}),$$

with $\mathbf{\Lambda}(\hat{\mathbf{U}}) = diag\{\hat{u} - c, \hat{u}, \hat{u}, \hat{u} + c\}$.

From the previous analysis, we know that the material derivative operator $\frac{D}{Dt} = \frac{\partial}{\partial t} + \hat{u}\frac{\partial}{\partial\hat{x}} + \hat{v}\frac{\partial}{\partial\hat{y}}$ plays an important role in our moving boundary treatment. Applying the material derivative operator to the boundary condition, we have

$$\frac{DT_b}{Dt} = 0, \quad \text{and} \quad \frac{D\mathbf{u}}{Dt} = \frac{D\mathbf{V}_b}{Dt}.$$

Here, $\frac{D\mathbf{V}_b}{Dt}$ is the acceleration of the rigid body. On the other hand, with the material derivative operator, the original NS equations can be written in the following form:

$$\begin{cases} \dfrac{D\rho}{Dt} + \rho(\hat{u}_{\hat{x}} + \hat{v}_{\hat{y}}) = 0, \\ \dfrac{D\hat{u}}{Dt} + \dfrac{1}{\rho}p_{\hat{x}} = \dfrac{1}{Re\rho}(\dfrac{4}{3}\hat{u}_{\hat{x}\hat{x}} + \hat{u}_{\hat{y}\hat{y}} + \dfrac{1}{3}\hat{v}_{\hat{x}\hat{y}}), \\ \dfrac{D\hat{v}}{Dt} + \dfrac{1}{\rho}p_{\hat{y}} = \dfrac{1}{Re\rho}(\hat{v}_{\hat{x}\hat{x}} + \dfrac{4}{3}\hat{v}_{\hat{y}\hat{y}} + \dfrac{1}{3}\hat{u}_{\hat{x}\hat{y}}), \\ \dfrac{DT}{Dt} + (\gamma - 1)T(\hat{u}_{\hat{x}} + \hat{v}_{\hat{y}}) = \dfrac{\gamma}{\rho\, Re\, Pr}(T_{\hat{x}\hat{x}} + T_{\hat{y}\hat{y}}) + NLT. \end{cases} \tag{3.14}$$

These would be used in our boundary treatment later.

Since the point value at the ghost point $P_{i,j}$ is obtained through a third order Taylor expansion in the $\hat{x}$-direction, approximations of the normal derivatives with suitable orders of accuracy are needed

$$\hat{\mathbf{U}}^{*(m)} \approx \partial_{\hat{x}}^m \hat{\mathbf{U}}|_{P_a}, \quad m = 0, 1, 2.$$

As before, we use the notations $(\cdot)_{ilw}$ and $(\cdot)_{ext}$ to denote derivatives obtained through the ILW procedure and the extrapolation respectively.

Firstly, we perform a local characteristic decomposition at P_a. However, different from the discussion in Section 2, we do not have the value of all components of $\hat{\mathbf{U}}$ on the boundary $\Gamma(t)$. Hence, we will use the extrapolation approximation $\hat{\mathbf{U}}^{*(0)}_{ext}$ in the characteristic decomposition. Let $\mathbf{V} = \mathbf{L}(\hat{\mathbf{U}}^{*(0)}_{ext})\hat{\mathbf{U}}$. We can see that the components $V_2, \cdots, V_4$ are the outflow variables, and V_1 is the inflow variable.

To construct $\hat{\mathbf{U}}^{*(0)}$, notice that we already know the value of $\hat{\mathbf{u}}$ and T at the boundary $\Gamma(t)$ given the boundary conditions, so we set

$$(\hat{U}_2)^{*(0)} = \hat{u}_b, \quad (\hat{U}_3)^{*(0)} = \hat{v}_b, \quad (\hat{U}_4)^{*(0)} = T_b. \tag{3.15}$$

Also, we can build the following relation since V_4 is outgoing,

$$\mathbf{l}_4(\hat{\mathbf{U}}^{*(0)}_{ext}) \cdot \hat{\mathbf{U}}^{*(0)} = V_4 = \mathbf{l}_4(\hat{\mathbf{U}}^{*(0)}_{ext}) \cdot \hat{\mathbf{U}}^{*(0)}_{ext}.$$

Solving this linear system, we can obtain $(\hat{U}_1)^{*(0)}$ as following,

$$(\hat{U}_1)^{*(0)} = (\hat{U}_1)^{*(0)}_{ext} \frac{2\,(\hat{U}_4)^{*(0)}_{ext} - (\hat{U}_4)^{*(0)} + \sqrt{\gamma \cdot (\hat{U}_4)^{(0)}_{ext}}\left((\hat{U}_2)^{*(0)}_{ext} - (\hat{U}_2)^{*(0)}\right)}{(\hat{U}_4)^{*(0)}_{ext}}. \tag{3.16}$$

Next, we try to find $\hat{\mathbf{U}}^{*(1)}$. Using the second equation in (3.14), we have

$$(\hat{U}_4)^{*(0)}(\hat{U}_1)^{*(1)}_{ilw} + (\hat{U}_1)^{*(0)}(\hat{U}_4)^{*(1)}_{ilw} = -(\hat{U}_1)^{*(0)}\frac{D\hat{u}_b}{Dt} + \frac{1}{Re}\left(\frac{4}{3}\hat{u}_{\hat{x}\hat{x}} + \hat{u}_{\hat{y}\hat{y}} + \frac{1}{3}\hat{v}_{\hat{x}\hat{y}}\right).$$

For the outflow variables $V_2, \ldots, V_4$, we have relations

$$\mathbf{l}_j \cdot \hat{\mathbf{U}}^{*(1)}_{ilw} = (V_j)^{*(1)}_{ext}, \quad j = 2, 3, 4.$$

Hence, we can get $\hat{\mathbf{U}}^{*(1)}_{ilw}$ by solving the above system. We apply combination (2.47) to get $\hat{\mathbf{U}}^{*(1)}$, which can be written as

$$\begin{aligned}
(\hat{V}_1)^{*(1)} &= \frac{c^2h^2}{c^2h^2 + 9\epsilon^2}(\hat{V}_1)^{*(1)}_{ilw} + \frac{9\epsilon^2}{c^2h^2 + 9\epsilon^2}(\hat{V}_1)^{*(1)}_{ext}, \\
(\hat{V}_j)^{*(1)} &= (\hat{V}_j)^{*(1)}_{ext} \quad j = 2, 3, 4,
\end{aligned}$$

with $\epsilon = \max\{\frac{1}{Re\rho}, \frac{4}{3Re\rho}, \frac{\gamma}{PrRe\rho}\}$ and $h = \sqrt{\Delta x^2 + \Delta y^2}$.

Notice that $(\hat{V}_j)^{*(1)}_{ilw} = (\hat{V}_j)^{*(1)}_{ext}, j = 2, 3, 4$ for the outflow characteristic variables. The second equation can be written as

$$(\hat{V}_j)^{*(1)} = (\hat{V}_j)^{*(1)}_{ext} = \frac{c^2h^2}{c^2h^2 + 9\epsilon^2}(\hat{V}_j)^{*(1)}_{ilw} + \frac{9\epsilon^2}{c^2h^2 + 9\epsilon^2}(\hat{V}_j)^{*(1)}_{ext} \quad j = 2, 3, 4.$$

Therefore, we set

$$\hat{\mathbf{U}}^{*(1)} = \frac{c^2h^2}{c^2h^2 + 9\epsilon^2}\hat{\mathbf{U}}^{*(1)}_{ilw} + \frac{9\epsilon^2}{c^2h^2 + 9\epsilon^2}\hat{\mathbf{U}}^{*(1)}_{ext}. \tag{3.17}$$

Finally, we will construct $\hat{\mathbf{U}}^{*(2)}$. According to the boundary conditions and (3.14), the second derivative $(\hat{U}_2)^{*(2)}_{ilw}, \ldots, (\hat{U}_4)^{*(2)}_{ilw}$ can be set as:

$$\begin{cases} (\hat{U}_2)^{*(2)}_{ilw} = \dfrac{3Re \cdot \rho}{4}\left(\dfrac{D\hat{u}_b}{Dt} + \dfrac{1}{\rho}p_{\hat{x}} - \dfrac{1}{Re \cdot \rho}(\hat{u}_{\hat{y}\hat{y}} + \dfrac{1}{3}\hat{v}_{\hat{x}\hat{y}})\right), \\ (\hat{U}_3)^{*(2)}_{ilw} = Re \cdot \rho\left(\dfrac{D\hat{v}_b}{Dt} + \dfrac{1}{\rho}p_{\hat{y}} - \dfrac{1}{Re \cdot \rho}(\dfrac{4}{3}\hat{v}_{\hat{y}\hat{y}} + \dfrac{1}{3}\hat{u}_{\hat{x}\hat{y}})\right), \\ (\hat{U}_4)^{*(2)}_{ilw} = \dfrac{Re \cdot \rho \cdot Pr}{\gamma}\left(\dfrac{DT_b}{Dt}(\gamma - 1)T(\hat{u}_{\hat{x}} + \hat{v}_{\hat{y}}) - \dfrac{\gamma}{Re \cdot \rho \cdot Pr}T_{\hat{y}\hat{y}} - NLT\right). \end{cases}$$

All the derivatives on the right hand side can be obtained by extrapolation. Note that, there is no diffusion term in the first equation of (3.14), so we just take

$$(\hat{U}_1)^{*(2)}_{ilw} = (\hat{U}_1)^{*(2)}_{ext}.$$

Finally, we combine $(\cdot)_{ilw}$ and $(\cdot)_{ext}$ as

$$(\hat{U}_j)^{*(2)} = \frac{c^2h^2}{c^2h^2 + 9\epsilon_j^2}(\hat{U}_j)^{*(2)}_{ext} + \frac{9\epsilon_j^2}{c^2h^2 + 9\epsilon_j^2}(\hat{U}_j)^{*(2)}_{ilw}, \tag{3.18}$$

where $\{\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\} = \{0, \frac{4}{3Re\rho}, \frac{1}{Re\rho}, \frac{\gamma}{PrRe\rho}\}$.

4 Numerical examples

In this section, we show some numerical examples to demonstrate that our method is stable and high order accurate if the solution is smooth. Our method also performs well for problems involving interactions between shocks and moving rigid bodies. We test all examples with the third order boundary treatment. The third order finite difference WENO approximation and fourth order central approximation are employed to discretize the convection terms and diffusion terms, respectively.

4.1 Accuracy tests

Example 1. We now consider the one-dimensional linear convection-diffusion equation

$$\begin{cases} u_t + cu_x = \epsilon u_{xx}, & x \in (x_l(t), x_r(t)),\ t > 0, \\ u(x_l(t), t) = g_l(t), & t > 0, \\ u(x_r(t), t) = g_r(t), & t > 0, \\ u(x, 0) = \sin x, & x \in (x_l(t), x_r(t)). \end{cases}$$

We take the left boundary as a continuous piecewise polynomial

$$x_l(t) = \begin{cases} t^4 - 2t^3 + \dfrac{3t^2}{2} - \dfrac{1}{2}, & 0 \leq t \leq \dfrac{1}{2} \\ \dfrac{1}{2}t - \dfrac{9}{16}, & t > \dfrac{1}{2}, \end{cases}$$

and the right moving boundary $x_r(t) = 0.5 + 0.5\sin^2(t)$. The functions $g_l(t)$ and $g_r(t)$ are given such that we have the exact solution $u(x,t) = e^{-\epsilon t}\sin(x - ct)$.

We take different pairs of (c, ϵ) to test the accuracy and stability of our algorithm for pure convection, convection-dominated, diffusion-dominated, and pure diffusion cases. In all situations, the time step is taken as

$$\Delta t = \min\left(\frac{0.6}{|c|/\Delta x + 6\epsilon/\Delta x^2}, \frac{\Delta x}{\max(|x_l'(t)|, |x_r'(t)|)}\right).$$

The numerical results at the final time $t_{end} = 1.0$ are listed in Table 1. Especially, for the cases of pure convection and convection-dominated, we set $c = 0.5$, such that the situation $f'(u(g_l(t))) - x_l'(t) = 0$ will occur during our computation when $t > \frac{1}{2}$. It can be seen from the table that the algorithm we have constructed is stable under different convection-diffusion coefficients and can achieve the expected third order accuracy, which demonstrates that our method performs well in all situations.

Table 1: Example 1: errors and convergence orders at $t_{end} = 1.0$.

Pure convection: $(c, \epsilon) = (0.5, 0)$				
h	L^1 error	order	L^∞ error	order
1/20	6.911E-006	–	4.879E-005	–
1/40	9.476E-007	2.866	1.042E-005	2.226
1/80	1.251E-007	2.920	2.141E-006	2.284
1/160	1.419E-008	3.139	3.633E-007	2.559
1/320	1.372E-009	3.370	5.401E-008	2.749
1/640	9.554E-011	3.844	5.828E-009	3.212
1/1280	4.676E-012	4.352	4.238E-010	3.781
Convection-dominated: $(c, \epsilon) = (0.5, 0.01)$				
h	L^1 error	order	L^∞ error	order
1/20	3.474E-006	–	1.604E-005	–
1/40	6.553E-007	2.406	5.049E-006	1.667
1/80	8.728E-008	2.908	4.939E-007	3.353
1/160	1.122E-008	2.958	7.195E-008	2.779
1/320	1.213E-009	3.209	9.803E-009	2.875
1/640	1.093E-010	3.472	1.310E-009	2.902
1/1280	1.013E-011	3.431	1.687E-010	2.957
Diffusion-dominated: $(c, \epsilon) = (0.01, 0.5)$				
h	L^1 error	order	L^∞ error	order
1/20	6.536E-006	–	3.540E-005	–
1/40	9.689E-007	2.753	4.172E-006	3.085
1/80	1.104E-007	3.132	4.769E-007	3.128
1/160	1.449E-008	2.930	9.401E-008	2.342
1/320	1.809E-009	3.001	8.270E-009	3.506
1/640	2.262E-010	2.999	1.454E-009	2.507
1/1280	2.903E-011	2.961	1.426E-010	3.349
Pure diffusion: $(c, \epsilon) = (0, 0.5)$				
h	L^1 error	order	L^∞ error	order
1/20	6.642E-006	–	3.542E-005	–
1/40	9.678E-007	2.778	4.168E-006	3.086
1/80	1.109E-007	3.124	4.787E-007	3.122
1/160	1.453E-008	2.932	9.411E-008	2.346
1/320	1.817E-009	2.999	8.285E-009	3.505
1/640	2.273E-010	2.998	1.456E-009	2.507
1/1280	2.850E-011	2.995	1.425E-010	3.353

Example 2. Next, we consider the following viscous Burgers' equation:

$$\begin{cases} u_t + uu_x = \epsilon\, u_{xx}, \quad x \in (x_l(t), x_r(t)),\, t > 0, \\ u(x_l(t), t) = g_l(t), \quad t > 0, \\ u(x_r(t), t) = g_r(t), \quad t > 0, \\ u(x, 0) = 0.5 - 0.5 \tanh((x - 0.5)/4\epsilon), \quad x \in (x_l(t), x_r(t)). \end{cases}$$

Boundary conditions $g_l(t)$ and $g_r(t)$ are given such that the exact solution is $u(x,t) = 0.5 - 0.5\tanh((x - 0.5 - 0.5t)/4\epsilon)$. The exact solution contains a sharp interface located at $x_r(t) = 0.5 + 0.5t$ for small ϵ. The time step is taken as

$$\Delta t = \min\left(\frac{0.6}{\max |u_j|/\Delta x + 6\epsilon/\Delta x^2}, \frac{\Delta x}{\max(|x_l'(t)|, |x_r'(t)|)}\right)$$

We simulate cases with different ϵ and moving boundary

$$x_l(t) = -0.5 + 0.5t, \quad x_r(t) = 0.5 + 0.5t,$$

or

$$x_l(t) = -0.5 + 0.5\sin^2(t), \quad x_r(t) = 0.5 + 0.5\sin^2(t).$$

The numerical results of $\epsilon = 1.0$ and 0.01 at the final time $t_{end} = 1.0$ are listed in Table 2. We can observe that the scheme achieves third order accuracy eventually with mesh refinements. Moreover, in Figure 2, we plot the numerical results of four different ϵ with $x_l(t) = -0.5 + 0.5t$, $x_r(t) = 0.5 + 0.5t$ and $h = 1/160$. We can see that a very sharp interface appears near boundary as ϵ goes to zero without obvious spurious oscillation, demonstrating the non-oscillatory property of our boundary treatment.

Example 3. Now, we test the 1D linear system

$$\begin{cases} \mathbf{U}_t + \mathbf{A}\mathbf{U}_x = \mathbf{B}\mathbf{U}_{xx} + \mathbf{S}, \quad x \in (x_l(t), x_r(t)),\, t > 0, \\ \mathbf{U}(x_l(t), t) = \mathbf{G}_l(t), \quad t > 0, \\ \mathbf{U}(x_r(t), t) = \mathbf{G}_r(t), \quad t > 0, \\ \mathbf{U}(x, 0) = \mathbf{U}_0(x), \quad x \in (x_l(t), x_r(t)). \end{cases}$$

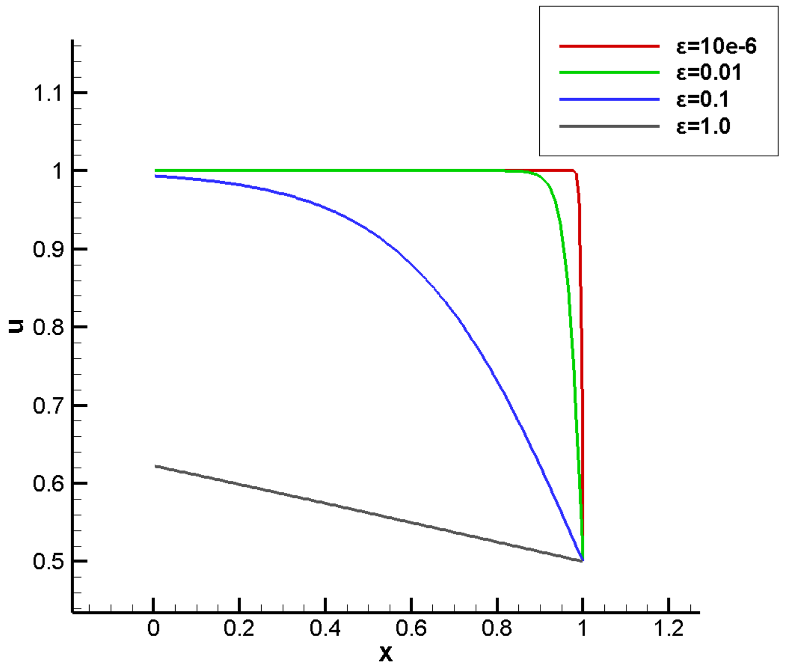


Figure 2: Example 2: Numerical solutions for different ϵ with $x_l(t) = -0.5 + 0.5t$, $x_r(t) = 0.5 + 0.5t$ and $h = 1/160$ at $t_{end} = 1.0$.

We consider the moving boundaries as $x_l(t) = -0.5+0.5t$ and $x_r(t) = 0.5+0.5t$. Here, $\mathbf{A}$ is a diagonalizable matrix and $\mathbf{B}$ is a diagonal matrix with positive elements. Specifically, we take

$$\mathbf{A} = \mathbf{A}_1 = \begin{pmatrix} 3 & 0.5 \\ 0.5 & 2 \end{pmatrix} \quad \text{or} \quad \mathbf{A} = \mathbf{A}_2 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 10^{-6} \end{pmatrix}, \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} b_{11} & 0 \\ 0 & b_{22} \end{pmatrix},$$

with $(b_{11}, b_{22}) = (0.8, 1)$, $(10^{-5}, 10^{-6})$ and $(0, 0)$, corresponding to the convection-diffusion case, convection-dominant case and pure convection case, respectively. Proper source term $\mathbf{S}$ is given such that exact solution is

$$\mathbf{U} = \begin{pmatrix} e^{-b_{11}t} \sin(x - a_{11}t) \\ e^{-b_{22}t} \cos(x - a_{22}t) \end{pmatrix}.$$

For this example, the time step is taken as

$$\Delta t = \min\left(\frac{0.6}{\sigma(A)/\Delta x + 6\sigma(B)/\Delta x^2}, \frac{\Delta x}{\max(|x_l'(t)|, |x_r'(t)|)} \right).$$

where the notation $\sigma(\cdot)$ means the spectral radius of the matrix. The numerical results

Table 2: Example 2: errors and convergence orders at $t_{end} = 1.0$

	$x_l(t) = -0.5 + 0.5t$, $x_r(t) = 0.5 + 0.5t$.							
	$\epsilon = 0.01$				$\epsilon = 1.0$			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
1/20	6.126E-003	–	0.111	–	1.510E-007	–	7.403E-007	–
1/40	1.889E-003	1.697	4.938E-002	1.169	1.792E-008	3.075	9.495E-008	2.962
1/80	4.431E-004	2.091	1.226E-002	2.009	2.191E-009	3.031	1.201E-008	2.982
1/160	7.377E-005	2.586	2.030E-003	2.595	2.696E-010	3.023	1.510E-009	2.991
1/320	9.893E-006	2.898	2.757E-004	2.880	3.328E-011	3.018	1.894E-010	2.995
1/640	1.195E-006	3.049	3.392E-005	3.023	4.154E-012	3.001	2.372E-011	2.997

	$x_l(t) = -0.5 + 0.5\sin^2(t)$, $x_r(t) = 0.5 + 0.5\sin^2(t)$.							
	$\epsilon = 0.01$				$\epsilon = 1.0$			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
1/20	2.593E-006	—	4.729E-005	–	1.521E-007	–	6.727E-007	–
1/40	7.065E-007	1.876	1.773E-005	1.414	1.801E-008	3.077	7.508E-008	3.163
1/80	2.219E-007	1.670	9.502E-006	0.900	2.076E-009	3.117	9.274E-009	3.017
1/160	5.637E-008	1.976	2.634E-006	1.850	2.366E-010	3.132	1.606E-009	2.528
1/320	9.503E-009	2.568	5.490E-007	2.262	3.206E-011	2.884	1.460E-010	3.460
1/640	1.355E-009	2.809	1.037E-007	2.403	3.811E-012	3.072	2.470E-011	2.563
1/1280	1.845E-010	2.877	1.236E-008	3.069	–	–	–	–

at $t_{end} = 1.0$ are listed in Table 3 and Table 4. We can observe the designed third order accuracy for all cases.

Example 4. Now we consider the Navier-Stokes equations with additional source terms so that we have an explicit exact solution to test accuracy. The modified system is

$$\begin{cases} \rho_t + (\rho u)_x = f_1(x,t), \\ (\rho u)_t + (\rho u^2 + p)_x = \dfrac{1}{Re}\left(\dfrac{4}{3}u\right)_{xx} + f_2(x,t), \\ E_t + (u(E+p))_x = \dfrac{1}{Re}\left(\dfrac{2}{3}(u^2)_{xx} + \dfrac{c^2}{(\gamma-1)Pr}\right) + f_3(x,t), \end{cases} \tag{4.1}$$

with $x_l(t) = -0.5 - 0.5t$ and $x_r(t) = 0.5 - 0.5t$. The exact solution is

$$\rho(x,t) = \frac{e^{\sin t}}{1+3x^2}, \quad u(x,t) = 1 + 3x^2, \quad p(x,t) = pre.$$

We consider the cases that $Re = 100$, 10^6 and $pre = 0.5$, 10 to demonstrate the effectiveness of our algorithm. We investigate the error of the numerical solutions with

Table 3: Example 3: errors and convergence orders at $t_{end} = 1.0$, for $\mathbf{A} = \mathbf{A}_1$.

Convection-diffusion case				
N	L^1 error	order	L^∞ error	order
20	3.405E-005	–	1.374E-004	–
40	4.868E-006	2.806	1.911E-005	2.847
80	6.533E-007	2.898	2.536E-006	2.913
160	8.477E-008	2.946	3.283E-007	2.950
320	1.078E-008	2.975	4.187E-008	2.971
640	1.357E-009	2.990	5.298E-009	2.983
1280	1.675E-010	3.018	6.672E-010	2.989
Convection-dominated case				
N	L^1 error	order	L^∞ error	order
20	8.887E-005	–	2.762E-004	–
40	1.092E-005	3.025	3.525E-005	2.970
80	1.374E-006	2.991	4.578E-006	2.945
160	1.708E-007	3.008	5.898E-007	2.957
320	2.130E-008	3.003	7.335E-008	3.007
640	2.656E-009	3.003	9.301E-009	2.979
1280	3.315E-010	3.002	1.157E-009	3.007
Pure convection case				
N	L^1 error	order	L^∞ error	order
20	8.887E-005	–	2.762E-004	–
40	1.092E-005	3.025	3.523E-005	2.971
80	1.371E-006	2.993	4.572E-006	2.946
160	1.705E-007	3.007	5.850E-007	2.966
320	2.135E-008	2.998	7.300E-008	3.002
640	2.671E-009	2.999	9.342E-009	2.966
1280	3.337E-010	3.000	1.156E-009	3.014

respect to the density ρ. The numerical results at $t_{end} = 1.0$ are listed in Table 5. We can observe third order accuracy for all the choices of Reynolds numbers and pressure. The time step is taken as

$$\Delta t = \min\left(\frac{0.6}{\lambda_c/\Delta x + \lambda_d/\Delta x^2}, \frac{\Delta x}{\max(|x_l'(t)|, |x_r'(t)|)}\right),$$

with $\lambda_c = \max\limits_j |u_j| + c$ and $\lambda_d = \max(\frac{1}{Re\,\rho}, \frac{4}{3Re\,\rho}, \frac{\gamma}{Pr\,Re\,\rho})$.

Table 4: Example 3: errors and convergence orders at $t_{end} = 1.0$, for $\mathbf{A} = \mathbf{A}_2$.

Convection-diffusion case				
N	L^1 error	order	L^∞ error	order
20	3.375E-005	–	1.627E-004	–
40	3.822E-006	3.142	2.132E-005	2.931
80	4.559E-007	3.067	2.731E-006	2.964
160	5.579E-008	3.030	3.455E-007	2.982
320	6.862E-009	3.023	4.345E-008	2.991
640	8.472E-010	3.017	5.448E-009	2.995
1280	1.053E-010	3.007	6.822E-010	2.997
Convection-dominated case				
N	L^1 error	order	L^∞ error	order
20	9.830E-005	–	2.892E-004	–
40	1.200E-005	3.033	3.962E-005	2.867
80	1.520E-006	2.981	5.218E-006	2.924
160	1.917E-007	2.987	6.681E-007	2.965
320	2.408E-008	2.992	8.368E-008	2.997
640	2.972E-009	3.018	1.029E-008	3.023
1280	3.336E-010	3.154	1.167E-009	3.140
Pure convection case				
N	L^1 error	order	L^∞ error	order
20	9.841E-005	–	2.896E-004	–
40	1.202E-005	3.032	3.972E-005	2.865
80	1.526E-006	2.978	5.248E-006	2.920
160	1.933E-007	2.981	6.726E-007	2.963
320	2.453E-008	2.978	8.589E-008	2.969
640	3.110E-009	2.979	1.082E-008	2.987
1280	3.930E-010	2.984	1.363E-009	2.989

Example 5. Now we consider the 2D linear convection-diffusion equation

$$\begin{cases} u_t + u_x + u_y = \epsilon(u_{xx} + u_{yy}), \quad (x, y) \in \Omega_i(t),\ t > 0, \\ u(x, y, t) = g(x, y, t), \quad (x, y) \in \partial\Omega_i(t),\ t > 0, \\ u(x, y, 0) = \sin x \sin y, \quad (x, y) \in \Omega_i(t), \end{cases}$$

Table 5: Example 4: errors and convergence orders at $t_{end} = 1.0$.

$Re = 100$								
	$pre = 0.5$				$pre = 10$			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
1/10	3.154E-003	–	1.972E-002	–	2.478E-003	–	9.677E-003	–
1/20	6.053E-004	2.381	2.578E-003	2.935	1.837E-004	3.753	1.139E-003	3.086
1/40	1.024E-004	2.562	4.245E-004	2.602	2.078E-005	3.144	1.213E-004	3.230
1/80	1.554E-005	2.721	5.607E-005	2.920	5.331E-006	1.963	2.918E-005	2.056
1/160	2.169E-006	2.840	7.163E-006	2.968	9.853E-007	2.435	5.185E-006	2.492
1/320	2.887E-007	2.909	9.518E-007	2.911	1.479E-007	2.735	7.300E-007	2.828
1/640	3.746E-008	2.946	1.256E-007	2.921	2.064E-008	2.840	9.317E-008	2.969
$Re = 10^6$								
	$pre = 0.5$				$pre = 10$			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
1/10	3.564E-003	–	1.922E-002	–	3.220E-003	–	1.172E-002	–
1/20	4.996E-004	2.834	2.777E-003	2.791	2.782E-004	3.532	1.263E-003	3.214
1/40	7.676E-005	2.702	7.178E-004	1.951	2.533E-005	3.457	1.185E-004	3.413
1/80	1.235E-005	2.635	1.128E-004	2.669	2.698E-006	3.230	1.222E-005	3.277
1/160	1.682E-006	2.875	2.052E-005	2.459	3.153E-007	3.097	1.472E-006	3.053
1/320	2.083E-007	3.013	3.078E-006	2.736	3.847E-008	3.034	1.854E-007	2.989
1/640	2.487E-008	3.066	3.003E-007	3.357	4.749E-009	3.018	2.330E-008	2.992

with the following three moving boundaries

$$\Omega_1(t) = \{(x,y)^T : (x-0.5t)^2 + (y-0.5t)^2 < 0.5\},$$

$$\Omega_2(t) = \{(x,y)^T : (x)^2 + (y)^2 < 0.5 - 0.2t\},$$

$$\Omega_3(t) = \{(x,y)^T : (x)^2 + (y)^2 < 0.5 + 0.2t\}.$$

Also, different diffusion coefficients $\epsilon = 0.1$ and 10^{-3} are considered. We give specific boundary conditions $g(x,y,t)$ so that the exact solution is $u(x,y,t) = e^{-2\epsilon t}\sin(x-t)\sin(y-t)$. The numerical results at $t_{end} = 1.0$ are listed in Table 6, with the time step

$$\Delta t = \min\left(\frac{0.6}{1/\Delta x + 1/\Delta y + \epsilon(1/\Delta x^2 + 1/\Delta y^2)},\ \frac{\Delta x}{\max\limits_{\partial\Omega}(|u_b|)}, \frac{\Delta y}{\max\limits_{\partial\Omega}(|v_b|)}\right).$$

Again, our scheme is stable and can achieve the designed third order accuracy for all cases.

Table 6: Example 5: errors and convergence orders at $t_{end} = 1.0$.

moving boundary $\Omega_1(t)$								
	$\epsilon = 0.1$				$\epsilon = 10^{-3}$			
N	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
20	2.344E-005	–	1.224E-004	–	1.209E-004	–	2.134E-003	–
40	3.891E-006	2.591	2.242E-005	2.448	8.540E-006	3.823	1.305E-004	4.031
80	5.856E-007	2.732	3.625E-006	2.629	7.341E-007	3.540	4.435E-006	4.878
160	7.536E-008	2.958	4.770E-007	2.925	7.063E-008	3.377	8.651E-007	2.358
320	9.711E-009	2.956	6.481E-008	2.879	4.112E-009	4.102	5.356E-008	4.013
moving boundary $\Omega_2(t)$								
	$\epsilon = 0.1$				$\epsilon = 10^{-3}$			
N	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
20	3.081E-005	–	2.613E-004	–	6.234E-005	–	1.124E-003	–
40	3.428E-006	3.168	2.841E-005	3.200	3.331E-006	4.226	9.606E-005	3.549
80	1.757E-007	4.285	3.064E-006	3.212	3.199E-007	3.380	3.916E-006	4.616
160	2.564E-008	2.777	4.033E-007	2.925	3.580E-008	3.159	6.672E-007	2.553
320	3.097E-009	3.049	5.502E-008	2.873	3.809E-009	3.232	6.652E-008	3.326
moving boundary $\Omega_3(t)$								
	$\epsilon = 0.1$				$\epsilon = 10^{-3}$			
N	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
20	2.233E-004	–	7.493E-004	–	3.124E-004	–	1.956E-003	–
40	8.014E-005	1.478	2.343E-004	1.677	4.029E-005	2.954	3.794E-004	2.366
80	1.907E-005	2.070	6.339E-005	1.886	6.524E-006	2.626	5.687E-005	2.737
160	4.968E-007	5.262	2.538E-006	4.642	2.170E-007	4.909	1.244E-006	5.513
320	3.821E-008	3.700	1.774E-007	3.838	1.519E-008	3.836	1.008E-007	3.624

4.2 Interaction between shocks and moving rigid bodies

In the following three examples, the time step Δt is taken as

$$\Delta t = \min\left(\frac{0.6}{\lambda_x/\Delta x + \lambda_y/\Delta y + \lambda_d(1/\Delta x^2 + 1/\Delta y^2)}, \frac{\Delta x}{\max(|u_b|)}, \frac{\Delta y}{\max(|v_b|)}\right),$$

with $\lambda_x = \max\limits_{ij} |u_{i,j}| + c$, $\lambda_y = \max\limits_{ij} |v_{i,j}| + c$, and $\lambda_d = \max(\frac{1}{Re\,\rho}, \frac{4}{3Re\,\rho}, \frac{\gamma}{Pr\,Re\,\rho})$.

Example 6. This example is the viscous flow version of Example 5 in [23] and Example 2 in [4]. We consider the interaction of shock wave and cylinder in a two-dimensional viscous fluid. The computational domain is $[0, 1] \times [0, 0.2]$. The horizontally moving shock wave is initially placed at the position of $x = 0.08$. A cylinder with radius $R_b = 0.05$ and surface density $\sigma = 10.77$ is immersed in the fluid, and the initial center position is $(0.15, 0.05)$. The initial state is shown in Figure 3, where, $\rho_R = 1.4$, $u_R = v_R = 0$ and

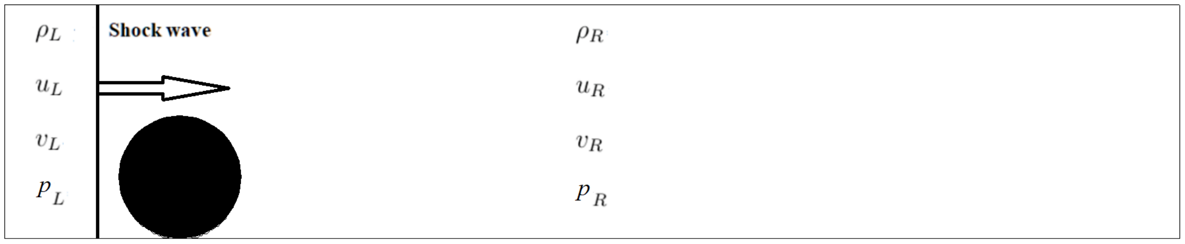


Figure 3: The initial state of Example 6.

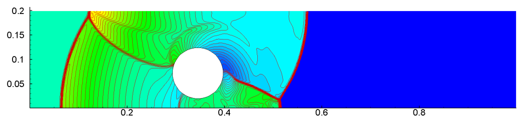
$p_R = 1.0$. For the left side of the initial shock wave, we set

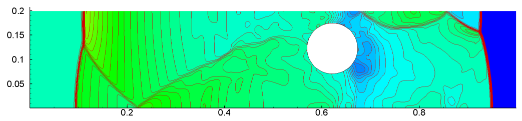
$$\begin{aligned}
\rho_L &= \frac{Ma_s^2 \gamma}{1. + (Ma_s^2 - 1)\frac{\gamma-1}{\gamma+1}}, \\
u_L &= \frac{2(Ma_s - \frac{1}{Ma_s})}{\gamma + 1}, \\
v_L &= 0, \\
p_L &= 1 + \frac{2\gamma(Ma_s^2 - 1)}{\gamma + 1},
\end{aligned}$$

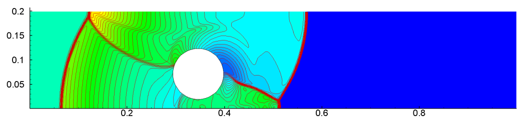
where Ma_s is the Mach number of the shock wave.

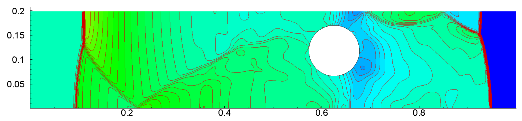
The governing equation is a dimensionless compressible Navier-Stokes equation. The left boundary and the right boundary of the computational domain are the inflow boundary and the outflow boundary, respectively. The upper and lower boundaries are reflective boundaries. The classical boundary treatments are applied on these outer lines. On the cylindrical wall $\Gamma(t)$, the isothermal no-slip wall boundary condition (3.6) is employed with the cylinder wall temperature $T_b = \frac{5}{7}$ (which is the temperature of the flow field at the right side of the initial shock). In our computation, $Pr = 0.7$ and $\gamma = 1.4$.

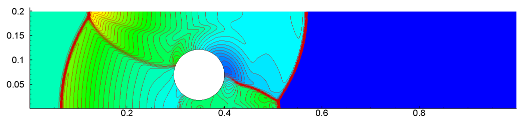
We test the problem with different Re and Mach number Ma_s under $\Delta x = \Delta y = \frac{1}{640}$. Pressure p and velocity magnitude $\|\mathbf{u}\|$ at different times are shown in Figure 4 - Figure 7. It is observed that our scheme is stable for all cases.


(a) Re=10^7, t=0.16410


(b) Re=10^7, t=0.30085


(c) Re=1000, t=0.16410


(d) Re=1000, t=0.30085


(e) Re=500, t=0.16410

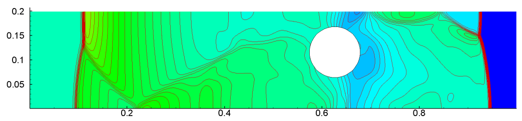
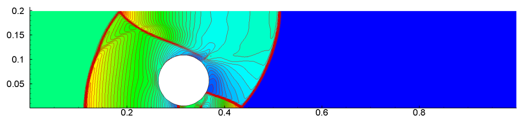
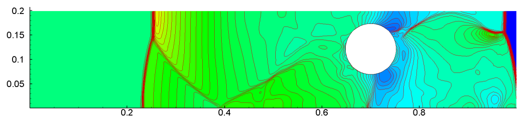
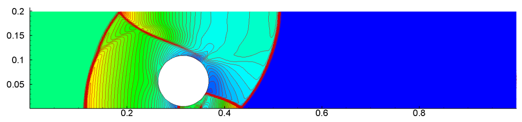
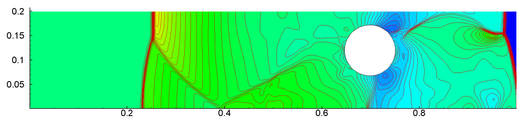

(f) Re=500, t=0.30085

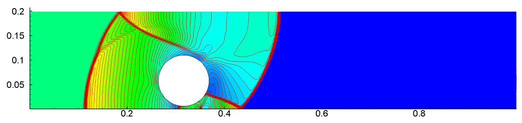
Figure 4: Example 6: Pressure contours (53 contours from 2 to 28), $Ma_s = 3$


(a) Re=10^7, t=0.07293


(b) Re=10^7, t=0.16410


(c) Re=1000, t=0.07293


(d) Re=1000, t=0.16410


(e) Re=500, t=0.07293

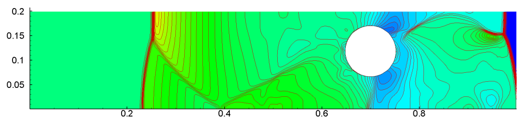
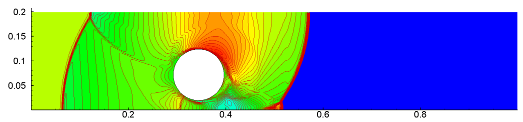

(f) Re=500, t=0.16410

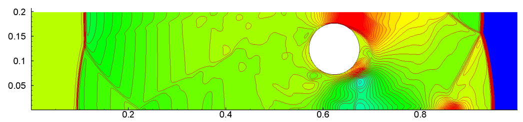
Figure 5: Example 6: Pressure contours (53 contours from 2 to 108), $Ma_s = 6$

For $Re = 1000$ and $Ma_s = 3$, the trajectory of the center under grids of different scales is shown in Figure 8. We can see that as the mesh is refined, the position of the center converges.

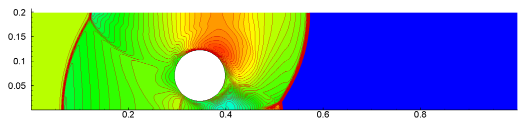
Example 7. Next, we replace the cylinder in Example 6 with a two-dimensional airfoil. Again, the computational domain is $[0, 1] \times [0, 0.2]$, and the grid width is $\Delta x = \Delta y = 1/1280$. The chord of the airfoil is $c_0 = 0.1$, and the area density is σ=10.77. The initial centroid position of the airfoil at the initial time is (0.15, 0.05), and the parameter



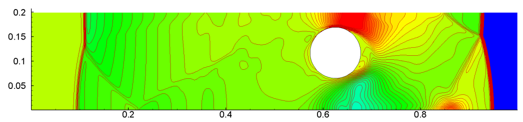
(a) Re=10^7, t=0.07293



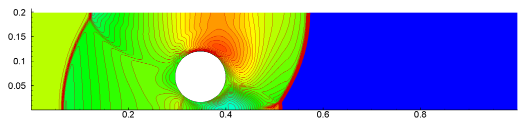
(b) Re=10^7, t=0.16410



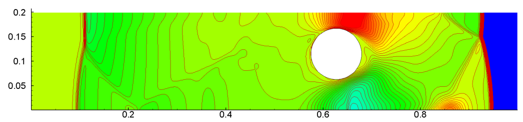
(c) Re=1000, t=0.07293



(d) Re=1000, t=0.16410

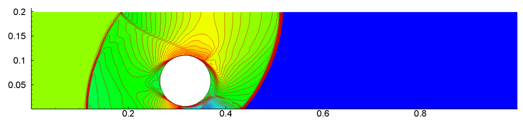


(e) Re=500, t=0.07293

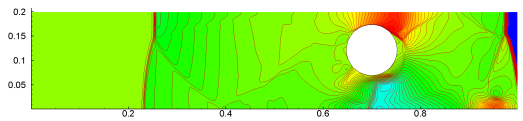


(f) Re=500, t=0.16410

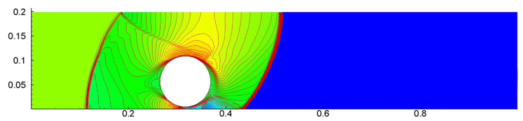
Figure 6: Example 6: Velocity magnitude contours (53 contours from 0.2 to 3.2), $Ma_s = 3$



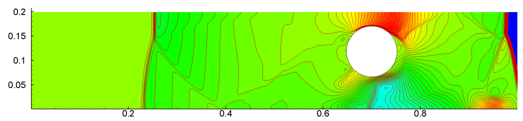
(a) Re=10^7, t=0.07293



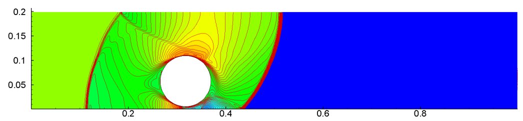
(b) Re=10^7, t=0.16410



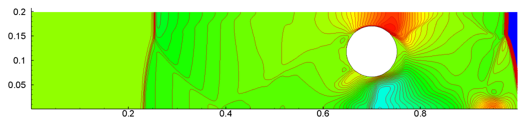
(c) Re=1000, t=0.07293



(d) Re=1000, t=0.16410



(e) Re=500, t=0.07293



(f) Re=500, t=0.16410

Figure 7: Example 6: Velocity magnitude contours (53 contours from 0.2 to 7.4), $Ma_s = 6$

equation of the airfoil is:

$$\begin{cases} x = s + 0.15 \\ y = f^{\pm}(s) + 0.05 \end{cases}$$

where $s \in [-a_0c_0, c_0 - a_0c_0]$ and

$$f^{\pm}(s) = \pm 0.6c_0(0.2969\sqrt{s_0} - 0.126s_0 - 0.3516s_0^2 + 0.28433s_0^3 - 0.1015s_0^4)$$

with $s_0 = \frac{s+a_0c_0}{c_0}$.

The same as in Example 6, the initial position of the shock wave is $x = 0.08$. The

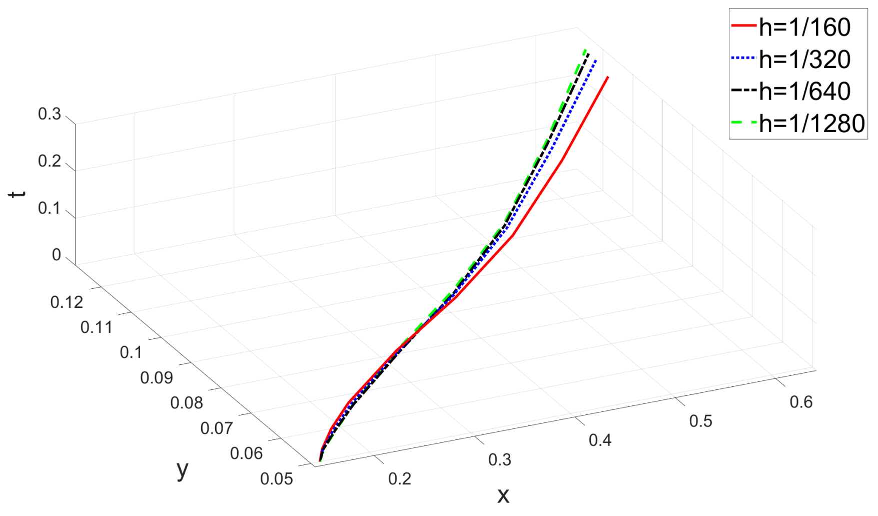


Figure 8: Example 6: The trajectory of the circle center calculated under the grid of $\Delta x = \Delta y = 1/160,\ 1/320,\ 1/640,\ 1/1280$.

numerical results at different times, different Re and Ma_s under the grid of $\Delta x = \Delta y = \frac{1}{640}$ are shown in Figure 9 - Figure 10, and the zoom view around the airfoil are shown in Figure 11 - Figure 12. No spurious oscillation appears around the surface of the airfoil. These indicate that our scheme can deal with the non-circle boundary well.

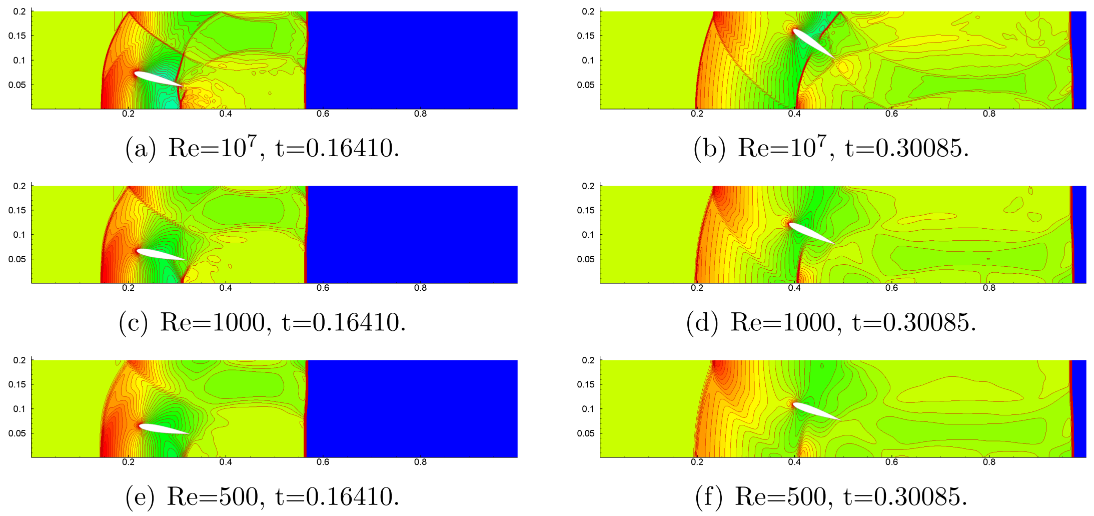
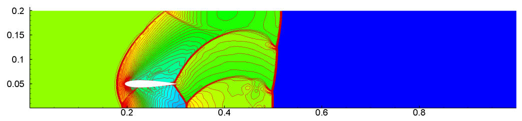
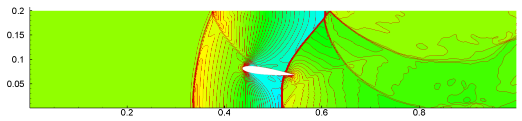
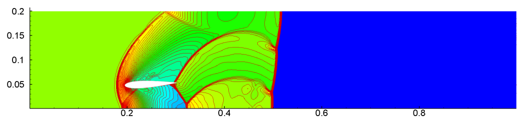

(a) Re=10^7, t=0.16410.
(b) Re=10^7, t=0.30085.
(c) Re=1000, t=0.16410.
(d) Re=1000, t=0.30085.
(e) Re=500, t=0.16410.
(f) Re=500, t=0.30085.

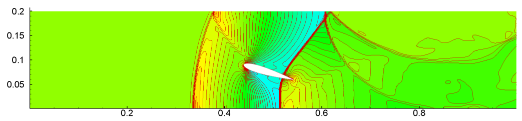
Figure 9: Example 7: Pressure contours (53 contours from 2 to 14),$Ma_s = 3$

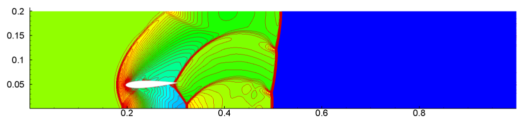
Example 8. Finally, we would like to simulate the Kármán vortex street. Consider the interaction between the shock wave and the cylinder. The computational domain


(a) Re=10^7, t=0.07293.


(b) Re=10^7, t=0.16410.


(c) Re=1000, t=0.07293.


(d) Re=1000, t=0.16410.


(e) Re=500, t=0.07293.

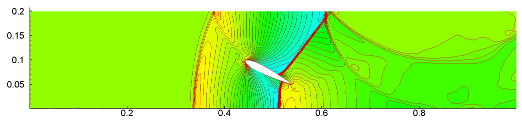

(f) Re=500, t=0.16410.

Figure 10: Example 7: Pressure contours (53 contours from 2 to 63),$Ma_s = 6$

is $[0, 4.0] \times [0, 1.0]$, and the mesh is divided into $\Delta x = \Delta y = 1/160$. The initial shock wave position is $x = 0.08$, with Mach number $Ma_s = 2.0$. We set the Reynolds number as $Re = 500$. The initial position of the center of the cylinder is $(0.2, 0.4)$, with radius $R_b = 0.05$, and the surface density is $\sigma = 5000$. The main difference from Example 7 is that we set upper and lower boundaries as adiabatic no-slip boundary. The numerical results at different moments are shown in Figure 13 and 14. We can see that our scheme can simulate the structures well.

5 Conclusion

In this paper we consider the numerical boundary conditions for high order finite difference schemes on Cartesian meshes to solve convection-diffusion equations in time-varying complex domains. Our method is an extension of the so-called inverse Lax-Wendroff procedure proposed in [14] for convection-diffusion equations in static geometries, in which a convex combination of boundary treatments for the diffusion-dominated and the convection-dominated cases was developed to obtain a stable and accurate boundary condition for general convection-diffusion equations. For moving boundaries, we convert material derivatives to spatial derivatives in the ILW procedure instead of using the Eulerian time derivatives in the ILW procedure in [14]. Moreover, our methodology gives a new definition of the weights to avoid zero denominator, such that the algorithm can be applied to solving pure convection and pure diffusion cases as well. To maintain high order accuracy in time, we employ the special time matching technique at the two intermediate Runge-Kutta stages. New treatment for the mixed derivatives at boundaries is designed to maintain high order accuracy and to reduce the computational cost. We also consider interactions between compressible viscous flows and moving rigid bodies. Numerical results show the high order accuracy and efficiency of our schemes. Our future work is to extend this method to convection-diffusion equations with Neumann boundary conditions.

References

[1] A. Baeza, P. Mulet and D. Zoro, *High order accurate extrapolation technique for finite difference methods on complex domains with Cartesian meshes*, Journal of Scientific Computing, 66 (2016), 761-791.

[2] M.J. Berger, C. Helzel and R.J. LeVeque, *h-box methods for the approximation of hyperbolic conservation laws on irregular grids*, SIAM Journal on Numerical Analysis, 41 (2003), 893-918.

[3] M.H. Carpenter, D. Gottlieb, S. Abarbanel and W.-S. Don, *The theoretical accuracy of Runge-Kutta time discretizations for the initial boundary value problem: a study of the boundary error*, SIAM Journal on Scientific Computing, 16 (1995), 1241-1252.

[4] Z. Cheng, S. Liu, Y. Jiang, J. Lu, M. Zhang and S. Zhang, *A high order boundary scheme to simulate a complex moving rigid body under the impingement of a shock wave*, Applied Mathematics and Mechanics (English Edition), 42 (2021), 841-854.

[5] S. Ding, C.-W. Shu and M. Zhang, *On the conservation of finite difference WENO schemes in non-rectangular domains using the inverse Lax-Wendroff boundary treatments*, Journal of Computational Physics, 415 (2020), 109516.

[6] F. Filbet and C. Yang, *An inverse Lax–Wendroff method for boundary conditions applied to Boltzmann type models*, Journal of Computational Physics, 245 (2013), 43-61.

[7] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, Journal of Computational Physics, 126 (1996), 202-228.

[8] H.-O. Kreiss and N.A. Petersson, *A second order accurate embedded boundary method for the wave equation with Dirichlet data*, SIAM Journal on Scientific Computing, 27 (2006), 1141-1167.

[9] H.-O. Kreiss, N.A. Petersson and J. Ystr̈om, *Difference approximations for the second order wave equation*, SIAM Journal on Numerical Analysis, 40 (2002), 1940-1967.

[10] H.-O. Kreiss, N. A. Petersson and J. Ystr̈om, *Difference approximations of the Neumann problem for the second order wave equation*, SIAM Journal on Numerical Analysis, 42 (2004), 1292-1323.

[11] T. Li, C.-W. Shu, and M. Zhang, *Stability analysis of the inverse Lax-Wendroff boundary treatment for high order upwind-biased finite difference schemes*, Journal of Computational and Applied Mathematics, 299 (2016), 140-158.

[12] T. Li, C.-W. Shu and M. Zhang, *Stability analysis of the inverse Lax-Wendroff boundary treatment for high order central difference schemes for diffusion equations*, Journal of Scientific Computing, 70 (2017), 576-607.

[13] T. Li, J. Lu and C.-W. Shu, *Stability analysis of inverse Lax-Wendroff boundary treatment of high order compact difference schemes for parabolic equations*, Journal of Computational and Applied Mathematics, 400 (2021), 113711.

[14] J. Lu, J. Fang, S. Tan, C.-W. Shu, M. Zhang, *Inverse Lax-Wendroff procedure for numerical boundary conditions of convection-diffusion equations*, Journal of Computational Physics, 317 (2016), 276-300.

[15] J. Lu, C.-W. Shu, S. Tan and M. Zhang, *An inverse Lax-Wendroff procedure for hyperbolic conservation laws with changing wind direction on the boundary*, Journal of Computational Physics, v426 (2021), 109940.

[16] R. Mittal and G. Iaccarino, *Immersed boundary methods*, Annual Review of Fluid Mechanics, 37 (2005), 239-261.

[17] S. Nilsson, N.A. Petersson, B. Sjögreen and H.-O. Kreiss, *Stable difference approximations for the elastic wave equation in second order formulation*, SIAM Journal on Numerical Analysis, 45 (2007), 1902-1936.

[18] C.S. Peskin, *The immersed boundary method*, Acta Numerica, 11 (2002), 1-39.

[19] Y. Qu, R. Shi and R.C. Batra, *An immersed boundary formulation for simulating high-speed compressible viscous flows with moving solids*, Journal of Computational Physics, 354 (2018), 672-691.

[20] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, Journal of Computational Physics, 77 (1988), 439-471.

[21] B. Sjögreen and N.A. Petersson, *A Cartesian embedded boundary method for hyperbolic conservation laws*, Communications in Computational Physics, 2 (2007), 1199-1219.

[22] S. Tan and C.-W. Shu, *Inverse Lax-Wendroff procedure for numerical boundary conditions of conservation laws*, Journal of Computational Physics, 229 (2010), 8144-8166.

[23] S. Tan and C.-W. Shu, *A high order moving boundary treatment for compressible inviscid flows*, Journal of Computational Physics, 230 (2011), 6023-6036.

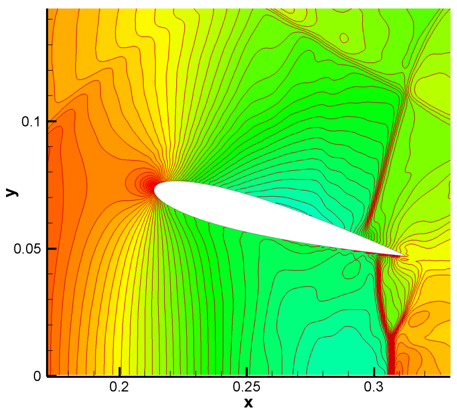
[24] S. Tan and C.-W. Shu, *Inverse Lax-Wendroff procedure for numerical boundary conditions of hyperbolic equations: survey and new developments*, in "Advances in Applied Mathematics, Modeling and Computational Science", R. Melnik and I. Kotsireas, Editors, Fields Institute Communications 66, Springer, New York, 2013, 41-63.

[25] S. Tan, C. Wang, C.-W. Shu and J. Ning, *Efficient implementation of high order inverse Lax-Wendroff boundary treatment for conservation laws*, Journal of Computational Physics, 231 (2012), 2510-2527.

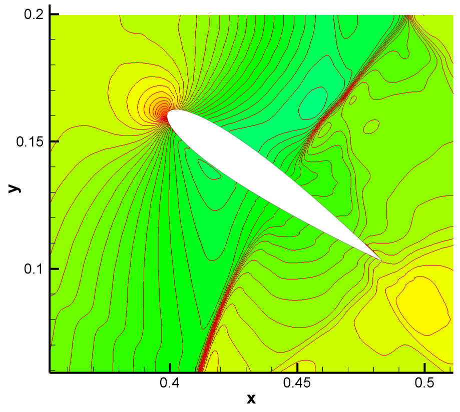
[26] F.D. Vanna, F. Picano, and E. Benini, *A sharp-interface immersed boundary method for moving objects in compressible viscous flows*, Computers & Fluids, 201 (2020), 104415.

[27] F. Vilar and C.-W. Shu, *Development and stability analysis of the inverse Lax-Wendroff boundary treatment for central compact schemes*, ESAIM: Mathematical Modelling and Numerical Analysis (M^2AN), 49 (2015), 39-67.

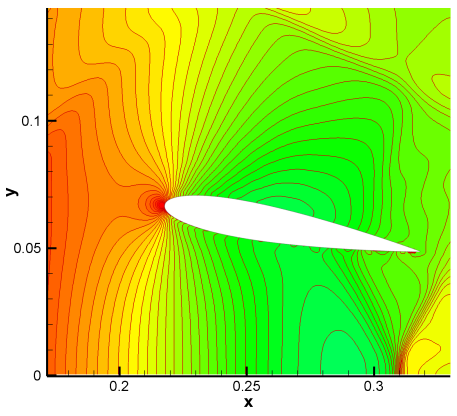
[28] J. Wang, X. Gu and J. Wu, *A sharp-interface immersed boundary method for simulating high-speed compressible inviscid flows*, Advances in Applied Mathematics and Mechanics, 12 (2020), 545-563.



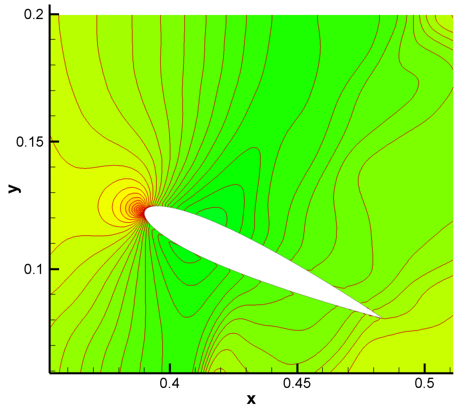
(a) Re=10^7, t=0.07293.



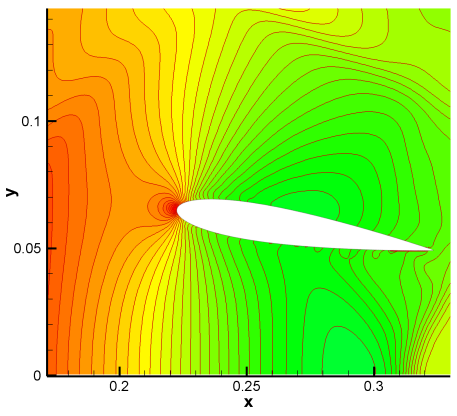
(b) Re=10^7, t=0.16410.



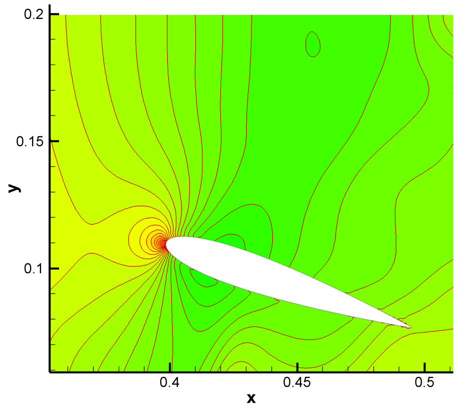
(c) Re=1000, t=0.07293.



(d) Re=1000, t=0.16410.

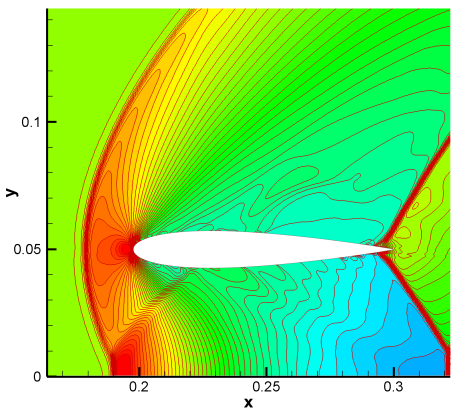


(e) Re=500, t=0.07293.

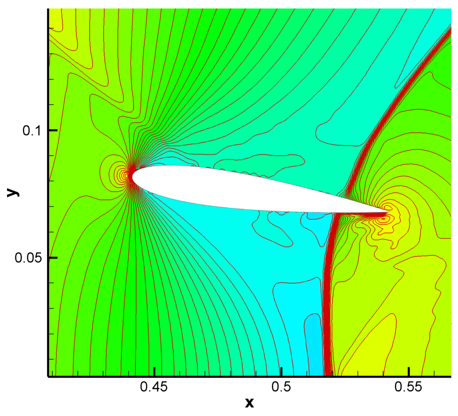


(f) Re=500, t=0.16410.

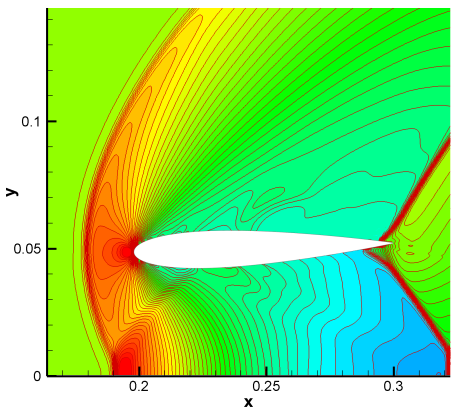
Figure 11: Example 7: Pressure contours around the airfoil (53 contours from 2 to 14),$Ma_s = 3$



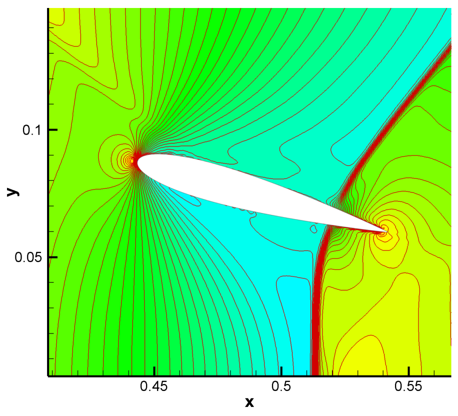
(a) Re=10^7, t=0.07293.



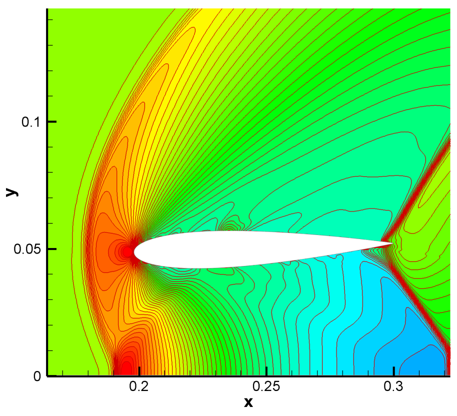
(b) Re=10^7, t=0.16410.



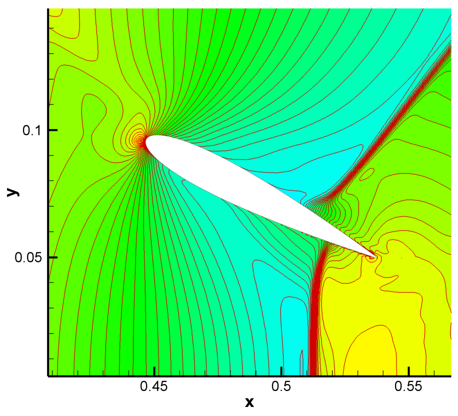
(c) Re=1000, t=0.07293.



(d) Re=1000, t=0.16410.

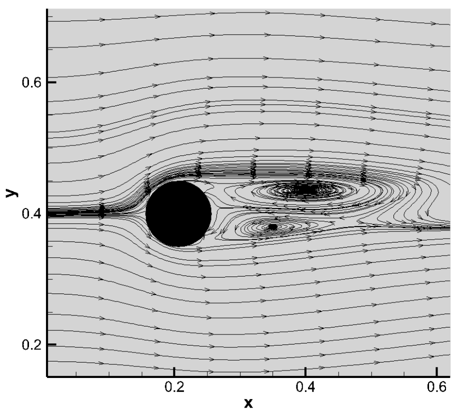


(e) Re=500, t=0.07293.

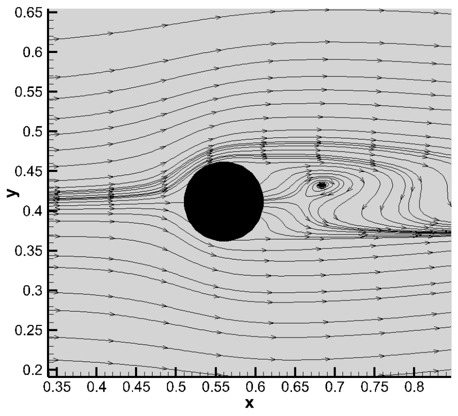


(f) Re=500, t=0.16410.

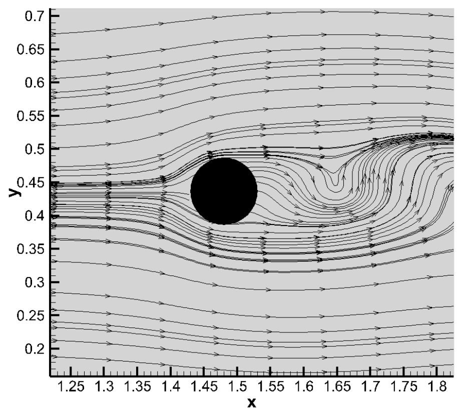
Figure 12: Example 7: Pressure contours around the airfoil (53 contours from 2 to 63),$Ma_s = 6$



(a) t=1.012

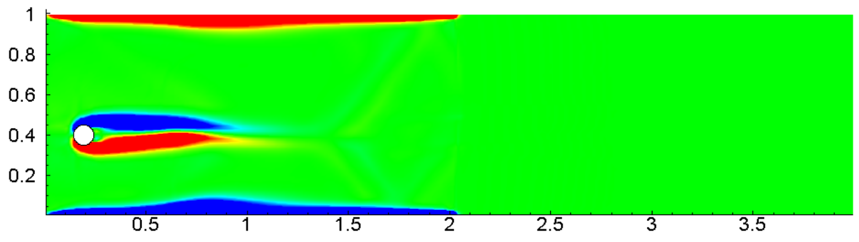


(b) t=7.988

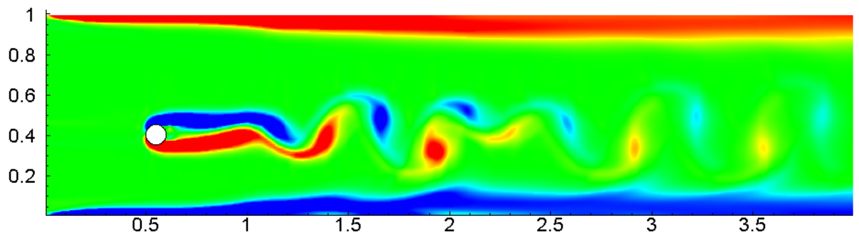


(c) t=15.5077

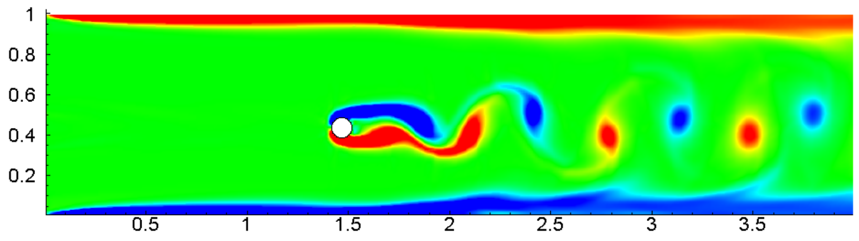
Figure 13: Example 8: Streamline at different moments



(a) t=1.012



(b) t=7.988



(c) t=15.5077

Figure 14: Example 8: The contour of vorticity