A Second-Order Finite-Difference Method for Compressible Fluids in Domains with Moving Boundaries

Alina Chertock [*], Armando Coco [†], Alexander Kurganov[‡] and Giovanni Russo[§]

Abstract

In this paper, we describe how to construct a finite-difference shock-capturing method for the numerical solution of the Euler equation of gas dynamics on arbitrary two-dimensional domain Ω, possibly with moving boundary. The boundaries of the domain are assumed to be changing due to the movement of solid objects/obstacles/walls. Although the motion of the boundary could be coupled with the fluid, all of the numerical tests are performed assuming that such a motion is prescribed and independent of the fluid flow. The method is based on discretizing the equation on a regular Cartesian grid in a rectangular domain $\Omega_R \supset \Omega$. We identify inner and ghost points. The inner points are the grid points located inside Ω, while the ghost points are the grid points that are outside Ω but have at least one neighbor inside Ω. The evolution equations for inner points data are obtained from the discretization of the governing equation, while the data at the ghost points are obtained by a suitable extrapolation of the primitive variables (density, velocities and pressure). Particular care is devoted to a proper description of the boundary conditions for both fixed and time dependent domains. Several numerical experiments are conducted to illustrate the validity of the method. We demonstrate that the second order of accuracy is numerically assessed on genuinely two-dimensional problems.

Key words: Compressible fluids, Euler equations of gas dynamics, ghost-cell extrapolation, moving boundaries, finite-difference shock-capturing methods.

AMS subject classifications: 76M20, 65M06, 65M20, 76N99.

1 Introduction

Development of accurate and efficient shock-capturing methods for hyperbolic systems of conservation laws has been a very active field of research in decades. Among the various methods,

[*]Department of Mathematics, North Carolina State University State University, Raleigh, NC 27695, USA; chertock@math.ncsu.edu

[†]Department of Mechanical Engineering and Mathematical Sciences, Oxford Brookes University, Oxford OX33 1HX, UK; acoco@brookes.ac.uk

[‡]Department of Mathematics, Southern University of Science and Technology of China, Shenzhen, 518055, China and Mathematics Department, Tulane University, New Orleans, LA 70118, USA; kurganov@math.tulane.edu

[§]Dipartimento di Matematica ed Informatica, Università di Catania, 95125, Catania, Italy; russo@dmi.unict.it

the ones that are most widely adopted are Discontinuous Galerkin (DG), finite-volume (FV) and conservative finite-difference (FD) schemes. DG and FV schemes are very flexible, since they can be naturally built on unstructured, possibly highly nonuniform meshes and thus can be applied on arbitrary domains. On the other hand, conservative FD schemes are easy to be made high-order in a very simple and efficient manner on Cartesian grid in several space dimensions. The main reason for their efficiency is that each flux derivative (converted into a flux difference along the coordinate directions) requires just one-dimensional (1-D) interpolations [30].

In view of the above considerations, our goal is to develop a simple and robust second-order conservative FD scheme for hyperbolic systems in the domains with fixed and moving boundaries. We concentrate on just one model—the Euler equations of gas dynamics. The methodology, however, could be also applied to other problems. In the two-dimensional (2-D) case, the governing equations are the compressible Euler equations:

$$\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ E\end{pmatrix}_t+\begin{pmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ u(E+p)\end{pmatrix}_x+\begin{pmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ v(E+p)\end{pmatrix}_y=\mathbf{0}, \tag{1.1}$$

where ρ is the density, u and v are the velocities, E is the total energy, and p is the pressure. The system (1.1) is closed using the equation of states (EOS), which in the case of a polytropic gas reads as

$$E=\frac{p}{\gamma-1}+\frac{\rho}{2}(u^2+v^2). \tag{1.2}$$

We also denote by $c:=\sqrt{\gamma p/\rho}$ the speed of sound, which will be used throughout the paper.

A simple FV method based on a Cartesian grid for the 1-D and 2-D compressible Euler equations in domains with solid moving boundaries was introduced in [8]. This FV method is an extension of the interface tracking method proposed in [7] for compressible multi-fluids. The key steps of the FV method from [8] are: (i) dividing all of the computational cells into the three groups: internal (fully occupied by the fluid), external (either located outside of the fluid domain or fully occupied by the solid obstacle), and boundary ones (partially filled by the fluid); (ii) evolving the solution in the interior cells only; (iii) replacing the unreliable data in the boundary cells with the data obtained by the solid wall extrapolation followed by the interpolation in the phase space. A slightly different approach was adopted in [18], in which arbitrary domains with moving boundaries were modeled using the level set method [25, 27] and a FV method on a Cartesian mesh for compressible Euler equations in one, two and three space dimensions was proposed.

One of the main advantages of the FV method proposed in [8] is its efficiency and robustness thanks to the lack of necessity to consider small internal cells near boundary (see, e.g., [4–6, 11, 21, 23]). However, the boundary representation in this interface tracking approach is only first-order accurate (as only the cells where the boundary is located were detected and no information on the exact location of the boundary was used). It should be observed that this drawback may be quite significant since moving boundaries/rigid objects typically generate waves of a large magnitude. Therefore, more accurate treatment of the moving solid wall boundary conditions is required to achieve a higher overall resolution.

In a recent work [2], a high-order shock-capturing FD method for hyperbolic conservation laws on arbitrary fixed 2-D domains was presented. In this method, the domain Ω is implicitly described by a level set function and the method is based on discretizing the equation on a regular Cartesian grid in a rectangular domain $\Omega_R \supset \Omega$. Inner points and ghost points are identified on the grid according to whether they are inside Ω or are outside Ω but have at least a neighbor inside it. The solution values at the inner points are obtained from the conservative FD discretization of the governing equation, while the values at the ghost points are obtained by a suitable extrapolation, which takes into account the boundary conditions on the field variables, when they are available.

In this paper, we develop a second-order conservative FD method for problems with both fixed and moving boundaries. In the case of fixed boundaries, our method is similar to the scheme from [2]. We, however, derive suitable boundary conditions for all field variables and use a different extrapolation technique, which is analogue to the one adopted in the context of elliptic [10] and elastic [9] problems. In particular, a 2-D interpolation which includes ghost points is used in order to impose boundary conditions on the projection of the ghost point values onto the boundary. A slightly modified procedure is derived in the case of moving boundaries, where the sets of inner and ghost points are time-dependent.

The paper is organized as follows. In §2 and §3, we describe the 1-D and 2-D numerical methods, respectively. We then test the proposed methods on several numerical examples reported in §4. In the test cases in which the solution is sufficiently smooth, we observe the second order of accuracy, while in the test cases with nonsmooth solutions, we demonstrate an advantage of the proposed boundary treatment by comparing it with a simple constant extrapolation.

2 One-Dimensional Numerical Method

We first consider the 1-D compressible Euler equations

$$\begin{pmatrix}\rho\\ \rho u\\ E\end{pmatrix}_t+\begin{pmatrix}\rho u\\ \rho u^2+p\\ u(E+p)\end{pmatrix}_x=\mathbf{0}, \tag{2.1}$$

which is closed with the 1-D version of the EOS (1.2):

$$E=\frac{p}{\gamma-1}+\frac{\rho u^2}{2},\quad \gamma=\text{Const.} \tag{2.2}$$

The system (2.1) can be written in the vector form

$$\boldsymbol{U}_t+\boldsymbol{f}(\boldsymbol{U})_x=\mathbf{0},\qquad \boldsymbol{U}:=\begin{pmatrix}\rho\\ \rho u\\ E\end{pmatrix},\quad \boldsymbol{f}(\boldsymbol{U}):=\begin{pmatrix}\rho u\\ \rho u^2+p\\ u(E+p)\end{pmatrix}.$$

In this section, we describe a numerical method for the system (2.1), (2.2) in the case of moving boundaries. We use a non-oscillatory second-order FD scheme from [28, 29, 31], which is briefly described in §2.1. To preserve the second order of accuracy, we introduce a special second-order boundary treatment procedure, see §2.2.

2.1 One-Dimensional Finite-Difference Scheme

We consider a uniform grid with $x_{j+1} - x_j = \Delta x$, $\forall j$. The point values $\boldsymbol{U}_j(t) :\approx \boldsymbol{U}(x_j, t)$ in the interior of the computational domain are then evolved in time using the following semi-discrete scheme:

$$\frac{d\boldsymbol{U}_j}{dt} = -\frac{\hat{\boldsymbol{f}}_{j+\frac{1}{2}} - \hat{\boldsymbol{f}}_{j-\frac{1}{2}}}{\Delta x}, \quad j = 1, \ldots, J. \tag{2.3}$$

The ODE system (2.3) is numerically integrated using the three-stage third-order strong stability preserving (SSP) Runge-Kutta method; see, e.g., [19, 20].

The numerical fluxes $\{\hat{\boldsymbol{f}}_{j+\frac{1}{2}}\}$ are computed as follows (see [28, 29, 31]). We first split the flux function $\boldsymbol{f}$ as

$$\boldsymbol{f}(\boldsymbol{U}_j) = \boldsymbol{f}_j^+ + \boldsymbol{f}_j^-, \qquad \boldsymbol{f}_j^\pm := \frac{1}{2}\big(\boldsymbol{f}(\boldsymbol{U}_j) \pm a_j \boldsymbol{U}_j\big), \tag{2.4}$$

where a_j is the spectral radius of the Jacobian matrix at $x = x_j$, which for compressible Euler equations is

$$a_j := |u_j| + c_j. \tag{2.5}$$

We then use a non-oscillatory generalized minmod reconstruction to evaluate the values of $\boldsymbol{f}^+$ and $\boldsymbol{f}^-$ at the cell interfaces $x_{j+\frac{1}{2}}$:

$$\boldsymbol{f}_j^{\rm E} := \boldsymbol{f}_j^+ + \frac{\Delta x}{2}(\boldsymbol{f}_x)_j^+, \qquad \boldsymbol{f}_j^{\rm W} := \boldsymbol{f}_j^- - \frac{\Delta x}{2}(\boldsymbol{f}_x)_j^-, \tag{2.6}$$

where the slopes are computed using the generalized minmod limiter (see, e.g., [22, 24, 32, 33]):

$$(\boldsymbol{f}_x)_j^\pm = \text{minmod}\left(\theta\,\frac{\boldsymbol{f}_j^\pm - \boldsymbol{f}_{j-1}^\pm}{\Delta x},\, \frac{\boldsymbol{f}_{j+1}^\pm - \boldsymbol{f}_{j-1}^\pm}{2\Delta x},\, \theta\,\frac{\boldsymbol{f}_{j+1}^\pm - \boldsymbol{f}_j^\pm}{\Delta x}\right), \tag{2.7}$$

where the minmod function is defined by

$$\text{minmod}(z_1, z_2, \ldots) := \begin{cases} \min(z_1, z_2, \ldots), & \text{if } z_i > 0 \ \ \forall i, \\ \max(z_1, z_2, \ldots), & \text{if } z_i < 0 \ \ \forall i, \\ 0, & \text{otherwise.} \end{cases} \tag{2.8}$$

Here, $\theta \in [1, 2]$ is a parameter that can control the amount of numerical dissipation: larger values of θ typically lead to sharper resolution of discontinuities, but may cause some oscillations. Finally, the numerical fluxes are given by

$$\hat{\boldsymbol{f}}_{j+\frac{1}{2}} = \boldsymbol{f}_j^{\rm E} + \boldsymbol{f}_{j+1}^{\rm W}. \tag{2.9}$$

Remark 2.1 The local choice of a_j in (2.5) may cause the numerical diffusion present in the scheme to be insufficient to prevent oscillations. The amount of numerical diffusion can be increased by replacing a_j with $\max_j a_j$.

Remark 2.2 The spatial accuracy of the scheme can be increased if one replaces the second-order minmod reconstruction (2.6)–(2.8) with a higher-order one such as ENO or WENO reconstructions (see, e.g., [28, 29, 31]).

Remark 2.3 Since we consider the case of moving boundaries, the set of interior points may change after one time step. Due to the CFL restriction, we may assume that the right-moving boundary, located at time $t + \Delta t$ between the points x_J and x_{J+1}, could be at time t either in the same interval $(x_J, x_{J+1}]$ or on the left in $(x_{J-1}, x_J]$ or on the right in $(x_{J+1}, x_{J+2}]$, see the schematic (x, t)-picture in Figure 2.1.

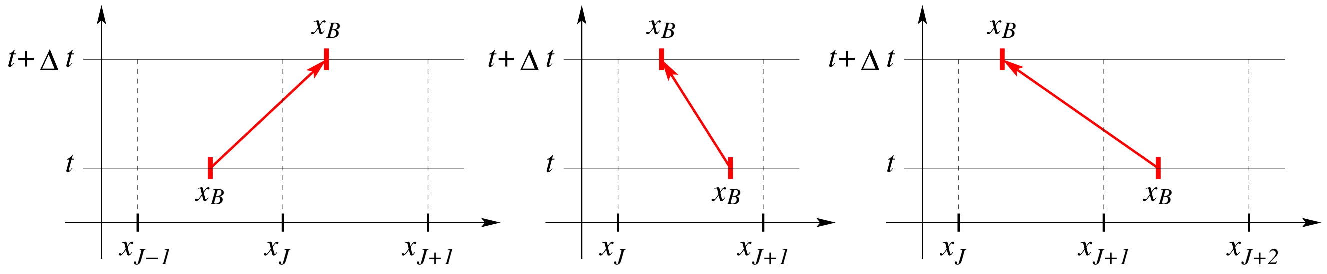

Figure 2.1: Three possible scenarios of the boundary (x_B) movements from time t to $t + \Delta t$: *Case 1* (center), *Case 2* (right) and *Case 3* (left).

Case 1: If $x_B(t) \in (x_J, x_{J+1}]$, then the ghost value $\boldsymbol{U}_{J+1}(t)$ should be computed using the boundary procedure described in §2.2.

Case 2: If $x_B(t) \in (x_{J+1}, x_{J+2}]$, then the value $\boldsymbol{U}_{J+1}(t)$ is available and no boundary extrapolation is needed.

Case 3: If $x_B(t) \in (x_{J-1}, x_J]$, then we first need to compute the ghost value $\boldsymbol{U}_J(t)$ and then to use the same boundary extrapolation procedure to obtain an additional ghost value $\boldsymbol{U}_{J+1}(t)$.

2.2 One-Dimensional Boundary Treatment

In this section, we will assume that only one ghost value on each side of the computational domain is needed to be able to use the scheme (2.3)–(2.9). Then, one has to specify how the values $\boldsymbol{U}_0$ and $\boldsymbol{U}_{J+1}$ at the ghost points x_0 and x_{J+1} are extrapolated from the interior values of $\boldsymbol{U}_j$, $j \in \{1, \dots, J\}$. Our goal is to impose (at least) second-order solid wall boundary conditions in the case of moving boundaries (it is well-known that the solid wall boundary conditions for non-moving boundaries are very easy to impose even for high-order schemes).

For the simplicity of the presentation, we consider the case in which the left boundary is fixed, while the right boundary is moving and its equation of motion $x_B = x_B(t)$ is prescribed (piston problem). In what follows, we will assume that $x_J < x_B(t + \Delta t) \le x_{J+1}$; see Figure 2.2.

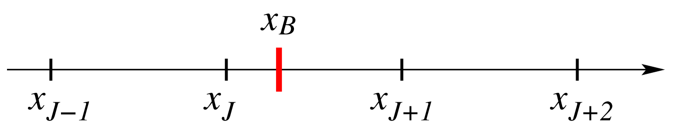

Figure 2.2: 1-D mesh: x_B represents the position of the right moving boundary.

As explained in §2.1, the FD method will require the values of u, p and ρ at the ghost points at time t in order to evolve the variables inside the domain. The boundary condition on the velocity is determined by the velocity of the right boundary so that we set $u(x_B, t) = x_B'(t)$. The pressure and density are extrapolated to the ghost points. This can be performed either in a naive way by extrapolating from internal to ghost points, or by considering additional conditions the pressure and density must satisfy at the boundary. In this paper, we follow the latter approach in which the required boundary conditions are obtained by imposing the compatibility with the governing equations of gas dynamics, as described in [16]. A comparison between the naive procedure and the proposed boundary treatment is performed in the 2-D numerical experiments; see §4.2.

In order to derive the solid wall boundary conditions at x_B, we first note that the Euler equations imply (for a smooth flow)

$$\rho\frac{Du}{Dt}+\frac{\partial p}{\partial x}=0 \quad\text{and}\quad \frac{DS}{Dt}=0,$$

where $D/Dt:=\partial/\partial t+u\partial/\partial x$ denotes the Lagrangian derivative and S is the entropy. Imposing that the first equation is valid near the wall gives a relation between the pressure gradient and the acceleration, while imposing that the space derivative of the entropy is zero near the wall (which is true for smooth solutions if it is true at the initial time; see [16] for details) gives a relation between the space derivatives of pressure and density. Summarizing, the three conditions at point x_B at any given time read as

$$\begin{cases} u(x_B)=x'_B,\\ \left.\dfrac{\partial p}{\partial x}\right|_{x=x_B}=-\rho(x_B)x''_B,\\ \left.\dfrac{\partial \rho}{\partial x}\right|_{x=x_B}=\dfrac{1}{c^2(x_B)}\cdot\left.\dfrac{\partial p}{\partial x}\right|_{x=x_B}=-\dfrac{\rho(x_B)}{c^2(x_B)}x''_B. \end{cases} \tag{2.10}$$

Observe that the condition on the pressure is a consequence of Newton's second law, while the condition on the density is obtained assuming that the boundary is adiabatic. To discretize the system (2.10), we approximate $u(x_B)$, $\rho(x_B)$ and $c^2(x_B)$ using an interpolation between the corresponding values at x_J and x_{J+1}. We also discretize the spatial derivatives using the FD approximation. The resulting discrete version of the system (2.10) is then

$$\begin{cases} \alpha u_{J+1}+(1-\alpha)u_J=x'_B,\\ \dfrac{p_{J+1}-p_J}{\Delta x}=-\left[\alpha\rho_{J+1}+(1-\alpha)\rho_J\right]x''_B,\\ \dfrac{\rho_{J+1}-\rho_J}{\Delta x}=-\dfrac{\alpha\rho_{J+1}+(1-\alpha)\rho_J}{\alpha c^2_{J+1}+(1-\alpha)c^2_J}\,x''_B, \end{cases} \tag{2.11}$$

where

$$\alpha:=\frac{x_B-x_J}{\Delta x}. \tag{2.12}$$

The system (2.11), (2.12) can be analytically solved for the unknowns ρ_{J+1}, u_{J+1} and p_{J+1}. Since the first equation is, in fact, decoupled from the other two equations, u_{J+1} can be computed directly, and the remaining equations form a 2×2 system for ρ_{J+1} and p_{J+1}, which can be solved explicitly. However, one can simplify the discrete boundary conditions even further by using simpler approximations of $\rho(x_B) \approx \rho_J$ and $c^2(x_B) \approx c_J^2$, which lead to the system

$$\begin{cases} \alpha u_{J+1} + (1-\alpha)u_J = x_B', \\ \dfrac{p_{J+1} - p_J}{\Delta x} = -\rho_J x_B'', \\ \dfrac{\rho_{J+1} - \rho_J}{\Delta x} = -\dfrac{\rho_J}{c_J^2} x_B'', \end{cases} \tag{2.13}$$

which can be explicitly solved to obtain the following boundary conditions:

$$u_{J+1} = \frac{x_B' - (1-\alpha)u_J}{\alpha}, \quad p_{J+1} = p_J - \Delta x \rho_J x_B'', \quad \rho_{J+1} = \rho_J \left[1 - \frac{\Delta x}{c_J^2} x_B''\right]. \tag{2.14}$$

The second-order numerical boundary conditions (2.11), (2.12) are, in fact, derived by using a linear interpolation between the corresponding point values at $x = x_J$ and $x = x_{J+1}$. For Dirichlet boundary conditions, this linear interpolation may lead to numerical instabilities for small values of α (that is, when x_B is close to x_J). To overcome this difficulty, we interpolate between the nodes $\{x_{J-1}, x_{J+1}\}$ instead of $\{x_J, x_{J+1}\}$ when α is below a fixed threshold $\alpha < \alpha^*$. In this case, the ghost value u_{J+1} is

$$u_{J+1} = \frac{x_B' - (1-\tilde{\alpha})u_{J-1}}{\tilde{\alpha}} \qquad \text{with} \quad \tilde{\alpha} = \frac{x_B - x_{J-1}}{2\Delta x} \geq \frac{1}{2}. \tag{2.15}$$

The value of α^* should be proportional to the mesh size; see, e.g., [2, 3, 17]. In our numerical experiments, we have used for simplicity $\alpha^* = 1/10$, which is five times larger than the biggest mesh size adopted ($\Delta x = \Delta y = 1/100$).

Remark 2.4 We notice that even though the boundary approximation (2.13) is by construction less accurate than (2.11), the boundary conditions (2.14), (2.12) remain second-order accurate as confirmed by the numerical experiments reported below.

Remark 2.5 To achieve a higher order of accuracy, say $2k > 2$, we need to use a $(2k)$-order FD method at the interior points, which will require k ghost points $x_{J+1}, \ldots, x_{J+k}$. For these points, one can use a higher-order polynomial interpolant using $2k$ grid points. In particular, for a ghost point x_{J+p}, with $1 \leq p \leq k$, one may use the point values at

$$\left\{x_{J-(2k-2)p}, x_{J-(2k-3)p}, \ldots, x_{J-2p}, x_{J-p}, x_J, x_{J+p}\right\}$$

to approximate the point values and spatial derivatives at $x = x_B$ in (2.10). For Dirichlet boundary conditions, if $\alpha < \alpha^*$ then this stencil can be replaced by

$$\left\{x_{J-(2k-2)p-1}, x_{J-(2k-3)p-1}, \ldots, x_{J-2p-1}, x_{J-p-1}, x_{J-1}, x_{J+p}\right\}.$$

In our paper, we only implement and test the second-order scheme, however we include this description for higher-order extensions.

Remark 2.6 In the 2-D case, the structure of ghost points is more complex and the equations analogous to the system (2.11), (2.12) are strongly coupled. Therefore, the 2-D boundary treatment procedure is much more complicated as we will see in §3.2.2.

3 Two-Dimensional Numerical Method

Let us suppose that the domain Ω is defined by a level set function $\phi(\boldsymbol{x})$, where $\boldsymbol{x} := (x,y)$, namely as the set $\{\boldsymbol{x}\in\mathbb{R}^2\colon \phi(\boldsymbol{x})<0\}$, while the obstacle is identified by $\{\boldsymbol{x}\in\mathbb{R}^2\colon \phi(\boldsymbol{x})>0\}$ and the boundary by $\Gamma := \{\boldsymbol{x}\in\mathbb{R}^2\colon \phi(\boldsymbol{x})=0\}$.

We observe that the level-set function ϕ that implicitly describes the obstacle is not unique, and that a particular level-set function is the *signed distance function*:

$$\phi(x,y)=\begin{cases}-\mathrm{dist}\big((x,y),\Gamma\big), & \text{if } (x,y)\in\Omega,\\ \mathrm{dist}\big((x,y),\Gamma\big), & \text{if } (x,y)\notin\Omega,\end{cases} \tag{3.1}$$

where $\mathrm{dist}\big((x,y),\Gamma\big)$ is the distance between the point (x,y) and the boundary Γ. The signed distance function of an obstacle can be obtained from a generic level-set function by adopting specific numerical methods, such as the reinitialization procedure; see, e.g., [15, 26].

Using the signed distance function (3.1), we identify the following three types of points (see Figure 3.1):

- Internal points with indices $(j,k)\in\mathcal{I}$, such that $\phi(x_j,y_k)<0$, that is, $(x_j,y_k)\in\Omega$. These are grid points at which the solution is evolved;
- Ghost points with indices $(j,k)\in\mathcal{G}$, such that $0\le\phi(x_j,y_k)<2h$. These are grid points located outside of Ω, but *near* the boundary. In particular, for a second-order method, the ghost points are within one or two grid cells (in either x- or y-direction) from the boundary.

  Within the ghost points, we distinguish between the first $\mathcal{L}_1$ $(0\le\phi(x_j,y_k)<h)$ and second $\mathcal{L}_2$ $(h\le\phi(x_j,y_k)<2h)$ layers $(\mathcal{L}_1\cup\mathcal{L}_2=\mathcal{G})$.

  In other words, the layer $\mathcal{L}_1$ is made by the ghost points located within one grid cell from the boundary, while the layer $\mathcal{L}_2$ is made of the ghost points located within two grid cells from the boundary that are not in the first layer; see Figure 3.1.
- Inactive points. These are grid points located outside of Ω, but not ghost points $(\phi(x_j,y_k)\ge 2h)$.

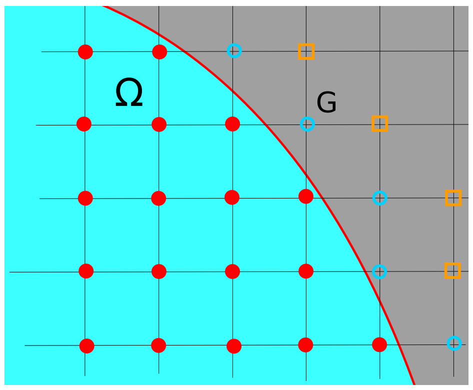

Figure 3.1: Grid points setup: The red filled circular points are inner points, the light blue unfilled circular and yellow unfilled square points are the first and second layers of ghost points, respectively.

Remark 3.1 The reinitialization step to obtain a signed distance function is fundamental for complex geometries. Our numerical experiments have been conducted in a relatively simple case of a (moving) disk, for which the signed distance function can be obtained explicitly:

$$\phi(x,y)=R-\sqrt{(x-x_c)^2+(y-y_c)^2},$$

where R is the radius and $(x_c,y_c)=(x_c(t),y_c(t))$ is the center of the disk.

3.1 Two-Dimensional Finite-Difference Scheme

We begin by rewriting the 2-D compressible Euler equations (1.1) in the vector form:

$$\boldsymbol{U}_t+\boldsymbol{f}(\boldsymbol{U})_x+\boldsymbol{g}(\boldsymbol{U})_y=\boldsymbol{0},\quad \boldsymbol{U}:=\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ E\end{pmatrix},\quad \boldsymbol{f}(\boldsymbol{U}):=\begin{pmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ u(E+p)\end{pmatrix},\quad \boldsymbol{g}(\boldsymbol{U}):=\begin{pmatrix}\rho u\\ \rho uv\\ \rho v^2+p\\ v(E+p)\end{pmatrix}.$$

As in the 1-D case, we consider a uniform grid with $x_{j+1}-x_j=\Delta x$, $y_{k+1}-y_k=\Delta y$, $\forall j,k$ and evolve the point values $\boldsymbol{U}_{j,k}(t):\approx\boldsymbol{U}(x_j,y_k,t)$ in the interior of the computational domain using the following semi-discrete FD scheme:

$$\frac{d\boldsymbol{U}_{j,k}}{dt}=-\frac{\hat{\boldsymbol{f}}_{j+\frac12,k}-\hat{\boldsymbol{f}}_{j-\frac12,k}}{\Delta x}-\frac{\hat{\boldsymbol{g}}_{j,k+\frac12}-\hat{\boldsymbol{g}}_{j,k-\frac12}}{\Delta y},\quad (j,k)\in\mathcal{I}. \tag{3.2}$$

The ODE system (3.2) is numerically integrated, as in the 1-D case, using the three-stage third-order SSP Runge-Kutta method. The numerical fluxes $\{\hat{\boldsymbol{f}}_{j+\frac12,k}\}$ and $\{\hat{\boldsymbol{g}}_{j,k+\frac12}\}$ are to be computed at interfaces between internal cells or between internal and first layer cells.

Following [28, 29, 31], we first split the flux functions $\boldsymbol{f}$ and $\boldsymbol{g}$ as

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{U}_{j,k})&=\boldsymbol{f}^+_{j,k}+\boldsymbol{f}^-_{j,k}, &\quad \boldsymbol{f}^\pm_{j,k}&:=\frac12\big(\boldsymbol{f}(\boldsymbol{U}_{j,k})\pm a_{j,k}\boldsymbol{U}_{j,k}\big),\\ \boldsymbol{g}(\boldsymbol{U}_{j,k})&=\boldsymbol{g}^+_{j,k}+\boldsymbol{g}^-_{j,k}, &\quad \boldsymbol{g}^\pm_{j,k}&:=\frac12\big(\boldsymbol{g}(\boldsymbol{U}_{j,k})\pm b_{j,k}\boldsymbol{U}_{j,k}\big),\end{aligned} \tag{3.3}$$

where $a_{j,k}$ and $b_{j,k}$ are the spectral radii of the Jacobian matrices $\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{U}}(\boldsymbol{U_{j,k}})$ and $\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{U}}(\boldsymbol{U_{j,k}})$, respectively, which for compressible Euler equations are

$$a_{j,k}:=|u_{j,k}|+c_{j,k},\quad b_{j,k}:=|v_{j,k}|+c_{j,k}. \tag{3.4}$$

We then use a non-oscillatory generalized minmod reconstruction to evaluate the values of $\boldsymbol{f}^\pm$ and $\boldsymbol{g}^\pm$ at the cell interfaces $(x_{j+\frac12},y_k)$ and $(x_j,y_{k+\frac12})$, respectively:

$$\begin{aligned}\boldsymbol{f}^{\rm E}_{j,k}&:=\boldsymbol{f}^+_{j,k}+\frac{\Delta x}{2}(\boldsymbol{f}_x)^+_{j,k}, &\quad \boldsymbol{f}^{\rm W}_{j,k}&:=\boldsymbol{f}^-_{j,k}-\frac{\Delta x}{2}(\boldsymbol{f}_x)^-_{j,k},\\ \boldsymbol{g}^{\rm N}_{j,k}&:=\boldsymbol{g}^+_{j,k}+\frac{\Delta y}{2}(\boldsymbol{g}_y)^+_{j,k}, &\quad \boldsymbol{g}^{\rm S}_{j,k}&:=\boldsymbol{g}^-_{j,k}-\frac{\Delta y}{2}(\boldsymbol{g}_y)^-_{j,k},\end{aligned} \tag{3.5}$$

where the slopes are computed using the generalized minmod limiter (see, e.g., [22, 24, 32, 33]):

$$\begin{aligned}
(\boldsymbol{f}_x)^{\pm}_{j,k} &= \mathrm{minmod}\left(\theta\,\frac{\boldsymbol{f}^{\pm}_{j,k}-\boldsymbol{f}^{\pm}_{j-1,k}}{\Delta x},\ \frac{\boldsymbol{f}^{\pm}_{j+1,k}-\boldsymbol{f}^{\pm}_{j-1,k}}{2\Delta x},\ \theta\,\frac{\boldsymbol{f}^{\pm}_{j+1,k}-\boldsymbol{f}^{\pm}_{j,k}}{\Delta x}\right),\\
(\boldsymbol{g}_y)^{\pm}_{j,k} &= \mathrm{minmod}\left(\theta\,\frac{\boldsymbol{g}^{\pm}_{j,k}-\boldsymbol{g}^{\pm}_{j,k-1}}{\Delta y},\ \frac{\boldsymbol{g}^{\pm}_{j,k+1}-\boldsymbol{g}^{\pm}_{j,k-1}}{2\Delta y},\ \theta\,\frac{\boldsymbol{g}^{\pm}_{j,k+1}-\boldsymbol{g}^{\pm}_{j,k}}{\Delta y}\right),
\end{aligned}\tag{3.6}$$

where the minmod function is defined by (2.8). Finally, the numerical fluxes are given by

$$\hat{\boldsymbol{f}}_{j+\frac{1}{2},k}=\boldsymbol{f}^{\mathrm{E}}_{j,k}+\boldsymbol{f}^{\mathrm{W}}_{j+1,k},\qquad \hat{\boldsymbol{g}}_{j,k+\frac{1}{2}}=\boldsymbol{g}^{\mathrm{N}}_{j,k}+\boldsymbol{g}^{\mathrm{S}}_{j,k+1}.\tag{3.7}$$

Remark 3.2 The local choices of $a_{j,k}$ and $b_{j,k}$ in (3.4) may cause the numerical diffusion present in the scheme to be insufficient to prevent oscillations. The amount of numerical diffusion can be increased by replacing $a_{j,k}$ and $b_{j,k}$ with $\max\limits_{j,k} a_{j,k}$ and $\max\limits_{j,k} b_{j,k}$, respectively.

Remark 3.3 The spatial accuracy of the scheme can be increased if one replaces the second-order minmod reconstruction (3.5), (3.6) with a higher-order one such as ENO or WENO reconstructions (see, e.g., [28, 29, 31]).

The computation of the values $\boldsymbol{U}_{j,k}$ at interior points $\big\{(x_j,y_k)\colon (j,k)\in\mathcal{I}\big\}$ according to (3.2)–(3.6) requires the information on the solution values at ghost points $\big\{(x_j,y_k)\colon (j,k)\in\mathcal{G}\big\}$ that arises from the discretization of the boundary conditions described below.

3.2 Two-Dimensional Boundary Treatment

In this section, we describe how the boundary conditions are set and discretized in both fixed and moving boundary cases.

We denote by $\boldsymbol{n}$ and $\boldsymbol{\tau}$ the normal and tangential unit vectors to the boundary $\partial\Omega$, respectively, and by κ the signed curvature of $\partial\Omega$ (see Figure 3.2). We assume that the unit normal points outside the fluid domain. We also denote by $\boldsymbol{u}:=(u,v)^T$ the velocity vector and by $u_n=\boldsymbol{u}\cdot\boldsymbol{n}$ and $u_\tau=\boldsymbol{u}\cdot\boldsymbol{\tau}$ its normal and tangential components, respectively.

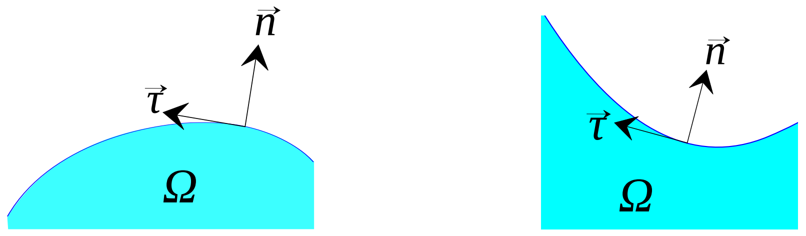

Figure 3.2: Locally convex ($\kappa<0$, left) and concave ($\kappa>0$, right) boundaries.

3.2.1 Fixed Boundary

We begin with a simpler case of a fixed (non moving) boundary and set four boundary conditions on the normal and tangential components of the velocity, pressure and density.

Condition on the normal component of the velocity. The condition on the normal velocity on the boundary $\partial\Omega$ is

$$u_n = 0. \tag{3.8}$$

Condition on the pressure. We first rewrite the momentum equations (the second and third equations in (1.1)) as

$$\rho \frac{D\boldsymbol{u}}{Dt} + \nabla p = \boldsymbol{0}, \tag{3.9}$$

where $D/Dt = \partial/\partial t + \boldsymbol{u} \cdot \nabla$ denotes the Lagrangian derivative.

The normal velocity condition (3.8) implies that along $\partial\Omega$ the velocity vector is

$$\boldsymbol{u} = u_\tau \boldsymbol{\tau}.$$

Therefore, taking into account the Frenet-Serret formulae we obtain

$$\frac{D\boldsymbol{u}}{Dt} = \frac{Du_\tau}{Dt}\boldsymbol{\tau} + u_\tau \frac{D\boldsymbol{\tau}}{Dt} = a_\tau \boldsymbol{\tau} + u_\tau^2 \kappa \boldsymbol{n}, \tag{3.10}$$

where a_τ is the tangential acceleration of the fluid and the curvature κ can be computed using the formula $|\kappa| = 1/R$, where R is the local radius of curvature of the boundary. With the notation in the Figure 3.2, κ is negative for locally convex regions and positive for locally concave ones.

We finally project equation (3.9) onto the normal direction and use (3.10) to obtain the boundary condition on the pressure:

$$\frac{\partial p}{\partial n} = -\rho u_\tau^2 \kappa. \tag{3.11}$$

Condition on the density. As in the 1-D case, the boundary condition on the density is obtained from the requirement that the boundary is adiabatic. More precisely, we assume that $\partial S/\partial n = 0$, where S denotes the entropy (this assumption has been widely used; see, e.g., [12–14]). This implies

$$\frac{\partial p/\partial n}{\partial \rho/\partial n} = \left.\frac{\partial p}{\partial \rho}\right|_{S=\text{Const}} = c^2 \quad \Longrightarrow \quad \frac{\partial \rho}{\partial n} = \left(\frac{\rho}{\gamma p}\right)\frac{\partial p}{\partial n}. \tag{3.12}$$

Condition on the tangential component of the velocity. To derive the boundary conditions on the tangential velocity, we consider the vorticity $\omega := v_x - u_y$ and compute it in the local coordinates $(\boldsymbol{n}, \boldsymbol{\tau})$:

$$\omega = \frac{\partial u_\tau}{\partial n} - \frac{\partial u_n}{\partial \tau} - u_\tau \kappa. \tag{3.13}$$

We then use (3.8) to obtain that $\partial u_n/\partial \tau = 0$ on the boundary, which reduces (3.13) to

$$\frac{\partial u_\tau}{\partial n} = u_\tau \kappa + \omega. \tag{3.14}$$

Finally, we assume that the flow is irrotational, which implies that $\omega = 0$ and then (3.14) becomes

$$\frac{\partial u_\tau}{\partial n} = u_\tau \kappa. \tag{3.15}$$

Remark 3.4 The boundary condition (3.15) can be also obtained by imposing the normal derivative of the total enthalpy density to be zero at the boundary. This condition has been used, for example, in [12–14].

In the case of a polytropic gas, the total enthalpy density is given by

$$h = \frac{\gamma}{\gamma-1}\cdot\frac{p}{\rho}+\frac{1}{2}|\boldsymbol{u}|^2, \tag{3.16}$$

and thus its normal derivative is

$$\frac{\partial h}{\partial n}=\frac{\gamma}{(\gamma-1)\rho}\left(\frac{\partial p}{\partial n}-\frac{p}{\rho}\frac{\partial\rho}{\partial n}\right)+\boldsymbol{u}\cdot\frac{\partial\boldsymbol{u}}{\partial n}.$$

Using the conditions (3.11) and (3.12), we simplify the previous expression to obtain

$$\frac{\partial h}{\partial n}=\boldsymbol{u}\cdot\frac{\partial\mathbf{u}}{\partial n}-u_\tau^2\kappa.$$

Finally, using (3.8) and the relation

$$\boldsymbol{u}\cdot\frac{\partial\boldsymbol{u}}{\partial n}=u_\tau\frac{\partial u_\tau}{\partial n}+u_n\frac{\partial u_n}{\partial n},$$

we arrive at the same boundary condition (3.15) on the tangential component of the velocity.

3.2.2 Discretization of the Fixed Boundary Conditions

In this section, we present a numerical discretization of the boundary conditions (3.8), (3.11), (3.12) and (3.15), derived in §3.2.1 for the case of fixed boundary. Notice that in this case we only need to compute the values of u, v, p and ρ at the ghost points in the first layer $\mathcal{L}_1$ of ghost points; see Figure 3.1.

Let us consider a ghost point $G\in\mathcal{L}_1$. We first compute the projection point $B\in\partial\Omega$, which is the closest point to G on the boundary (see Figure 3.3). To this end, we use the signed distance function (3.1), that is,

$$B\equiv(x_B,y_B)=G-\phi(G)\,\frac{\nabla\phi(G)}{|\nabla\phi(G)|}. \tag{3.17}$$

We then define a 3×3 stencil $\mathcal{S}_G$ also outlined in Figure 3.3. Note that this stencil consists of ghost and interior points only and that the boundary point B is contained within $\mathcal{S}_G$.

Let us denote by $\mathcal{Q}[\psi;\mathcal{S}]$ the bi-quadratic interpolant of a grid function $\{\psi_{j,k}\}$ in the 3×3 stencil $\mathcal{S}$. We recall that the bi-quadratic interpolant is the polynomial

$$\mathcal{Q}[\psi;\mathcal{S}](x,y)=\sum_{m,n=0}^{2}a_{mn}x^my^n,$$

such that $\mathcal{Q}[\psi;\mathcal{S}](x_j,y_k)=\psi_{j,k}$ for any point (x_j,y_k) of the stencil $\mathcal{S}$. Then, the discretization of the boundary conditions is obtained by approximating all of the values at point B in (3.8), (3.11),

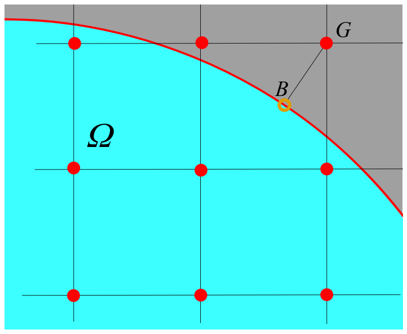

Figure 3.3: Ghost point $G \in \mathcal{L}_1$, its boundary projection $B \in \partial\Omega$, and the stencil $\mathcal{S}_G$ (red filled circular points).

(3.12) and (3.15) using the corresponding bi-quadratic interpolants:

$$\begin{aligned}
&\mathcal{Q}[u_n;\mathcal{S}_G](B) = 0,\\
&\frac{\partial \mathcal{Q}[u_\tau;\mathcal{S}_G](B)}{\partial n} = \kappa \mathcal{Q}[u_\tau;\mathcal{S}_G](B),\\
&\frac{\partial \mathcal{Q}[p;\mathcal{S}_G](B)}{\partial n} = -\kappa \mathcal{Q}[\rho;\mathcal{S}_G](B)\big(\mathcal{Q}[u_\tau;\mathcal{S}_G](B)\big)^2,\\
&\frac{\partial \mathcal{Q}[\rho;\mathcal{S}_G](B)}{\partial n} = \frac{\mathcal{Q}[\rho;\mathcal{S}_G](B)}{\gamma \mathcal{Q}[p;\mathcal{S}_G](B)} \cdot \frac{\partial \mathcal{Q}[p;\mathcal{S}_G](B)}{\partial n}.
\end{aligned} \tag{3.18}$$

It can be shown that the computation of $(u_n)_G$ through the bi-quadratic interpolation $\mathcal{Q}[u_n;\mathcal{S}_G](B)$ leads to a denominator of $\alpha_x\alpha_y(1+\alpha_x)(1+\alpha_y)/4$, where (see Figure 3.4)

$$\alpha_x = \frac{|x_B - x_P|}{\Delta x} \quad \text{and} \quad \alpha_y = \frac{|y_B - y_P|}{\Delta y} \quad \text{with} \quad B := (x_B, y_B) \;\; \text{and} \;\; P := (x_P, y_P),$$

and P is the internal grid point closest to B. As in the 1-D case, this may produce numerical instability when at least one of α_x and α_y is small. If so, we avoid small denominators by modifying the stencil as follows. If α_x is below a prescribed threshold α_x^*, then the stencil is enlarged in the x-direction, as shown in Figure 3.4. If α_y is below a prescribed threshold α_y^*, then the stencil is modified analogously in the y-direction. Note that if both α_x and α_y are below their respective thresholds, then the stencil is modified in both directions. Following the same argument of the 1-D case, in our numerical experiments we have used $\alpha_x^* = \alpha_y^* = 1/10$.

We now note that unlike the 1-D case, we cannot solve separately the 4×4 systems obtained from (3.18) for each ghost point G, since each 4×4 system may be coupled with the corresponding 4×4 systems obtained at other ghost points. We therefore adopt an iterative technique that was successfully employed in the context of elliptic equations in [10]. In particular, we transform the system of boundary conditions (3.8), (3.11), (3.12) and (3.15) into a time-dependent problem with

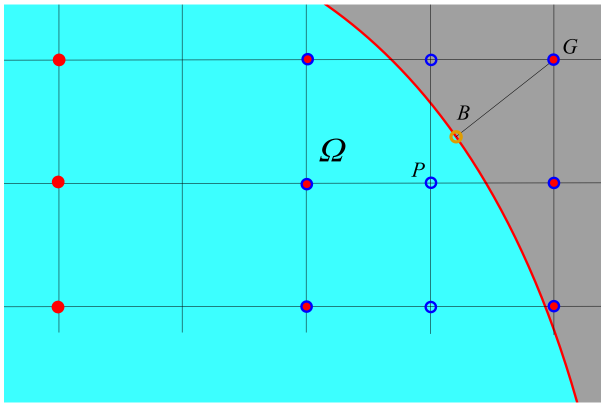

Figure 3.4: Enlarged stencil $\mathcal{S}_G$ for the ghost point G: since $\alpha_x = \frac{|x_B - x_P|}{\Delta x}$ is below a prefixed threshold, the original stencil (blue unfilled circular points) is enlarged in the x-direction (red filled circular points).

a fictitious time σ:

$$\begin{aligned}
\frac{\partial u_n}{\partial \sigma} &= -u_n,\\
\frac{\partial u_\tau}{\partial \sigma} &= -\mu_2\left(\frac{\partial u_\tau}{\partial n} - u_\tau\kappa\right),\\
\frac{\partial p}{\partial \sigma} &= -\mu_1\left(\frac{\partial p}{\partial n} + \rho u_\tau^2\kappa\right),\\
\frac{\partial \rho}{\partial \sigma} &= -\mu_3\left(\frac{\partial \rho}{\partial n} - \frac{1}{c_s^2}\frac{\partial p}{\partial n}\right),
\end{aligned} \tag{3.19}$$

where μ_1, μ_2 and μ_3 are suitable constants, which will be chosen below. The iterative scheme is then obtained by discretizing (3.19) in space and time (the fictitious time σ represents an iterative parameter) and computing its steady-state solution. The partial derivatives with respect to σ are discretized at the ghost points G using the first-order forward Euler method (this is good enough to reach the steady state), while all the other terms are discretized at the corresponding boundary points B using the same second-order discretization as in (3.18). The resulting iterative scheme reads as

$$\begin{aligned}
&u_n^{(m+1)}(G) = u_n^{(m)}(G) - \Delta\sigma\mathcal{Q}[u_n^{(m)};\mathcal{S}_G](B),\\
&u_\tau^{(m+1)}(G) = u_\tau^{(m)}(G) - \mu_2\Delta\sigma\left(\frac{\partial\mathcal{Q}[u_\tau^{(m)};\mathcal{S}_G](B)}{\partial n} - \kappa\mathcal{Q}[u_\tau^{(m)};\mathcal{S}_G](B)\right),\\
&p^{(m+1)}(G) = p^{(m)}(G) - \mu_1\Delta\sigma\left(\frac{\partial\mathcal{Q}[p^{(m)};\mathcal{S}_G](B)}{\partial n} + \kappa\mathcal{Q}[\rho^{(m)};\mathcal{S}_G](B)\big(\mathcal{Q}[u_\tau^{(m)};\mathcal{S}_G](B)\big)^2\right),\\
&\rho^{(m+1)}(G) = \rho^{(m)}(G) - \mu_3\Delta\sigma\left(\frac{\partial\mathcal{Q}[\rho^{(m)};\mathcal{S}_G](B)}{\partial n} - \frac{\mathcal{Q}[\rho^{(m)};\mathcal{S}_G](B)}{\gamma\mathcal{Q}[p^{(m)};\mathcal{S}_G](B)}\cdot\frac{\partial\mathcal{Q}[p^{(m)};\mathcal{S}_G](B)}{\partial n}\right).
\end{aligned} \tag{3.20}$$

The fictitious time step $\Delta\sigma$ and the constants μ_i, $i = 1, 2, 3$ are chosen in order to satisfy the CFL conditions for (3.20), namely we take:

$$\Delta\sigma < 1 \quad \text{and} \quad \mu_i\Delta\sigma < \min(\Delta x, \Delta y), \quad i = 1, 2, 3.$$

Iterations (3.20) are performed until the residual falls below a prescribed tolerance.

Remark 3.5 In all of the numerical examples presented below, the computational time devoted to the solution in the iterative process is a small fraction of the total computational time. However, in more complicated geometries, a direct implementation of the iterative scheme (3.20) may lead to slow convergence. It would be therefore convenient to apply a block-relaxation approach, which takes advantage of the fact that the systems (3.20) for different ghost points are not always fully coupled, or Newton's iteration scheme, which would lead to even faster convergence. Moreover, in a special case of an obstacle being a rigid disk, the systems are even fully decoupled. In fact, it can be easily proven that for any $G \in \mathcal{L}_1$, the stencil $\mathcal{S}_G$ may only involve the ghost points, which are closer to the interface than G. Then, if we order the ghost points according to the distance from the interface and perform the iteration from the closest to the farthest point in a Gauss-Seidel fashion (obtained by using a suitable choice of $\Delta\sigma$ and μ_i's in (3.20)), the iterations would convergence in one sweep only, but even without such an ordering of the ghost points, the scheme converges in just few iterations.

3.2.3 Moving Boundary

In this section, we extend the boundary conditions presented in §3.2.1 from the case of a fixed boundary to the case of a moving boundary. As before, we assume that the boundary is adiabatic and the flow is irrotational.

Let us assume that the solid obstacle/boundary is moving according to a prescribed equation of motion and that the normal boundary velocity is a given function $V(\boldsymbol{x},t)$. In this case, the boundary can be represented by the signed distance function $\phi(\boldsymbol{x},t)$ that satisfies the following PDE:

$$\frac{\partial \phi}{\partial t} + V(\boldsymbol{x},t)\,|\nabla\phi| = 0.$$

Then, the normal boundary velocity $V(\boldsymbol{x},t)$ can be expressed by

$$V(\boldsymbol{x},t) = -\frac{\partial \phi}{\partial t}.$$

For example, if the moving obstacle is a disk of radius R centered at $\boldsymbol{c}(t) \equiv (x_c(t), y_c(t))$ and moving with the given velocity $\boldsymbol{u_c}(t)$, then one has

$$\phi(\boldsymbol{x},t) = R - |\boldsymbol{c}(t) - \boldsymbol{x}| \quad \Longrightarrow \quad V(\boldsymbol{x},t) = \frac{\boldsymbol{c}(t)-\boldsymbol{x}}{|\boldsymbol{c}(t)-\boldsymbol{x}|}\cdot \boldsymbol{u_c(t)} = \boldsymbol{n}\cdot\boldsymbol{u}_c(t).$$

Condition on the normal component of the velocity. In the case of a moving boundary, the boundary conditions on $\partial\Omega$ for the normal velocity becomes

$$u_n = V. \tag{3.21}$$

Condition on the pressure. The condition is obtained as follows. The normal velocity condition (3.21) implies that along $\partial\Omega$ the velocity vector is

$$\boldsymbol{u} = u_\tau \boldsymbol{\tau} + V\boldsymbol{n}. \tag{3.22}$$

The Lagrangian derivative of the velocity is given by

$$\frac{D\boldsymbol{u}}{Dt}=\frac{Du_\tau}{Dt}\boldsymbol{\tau}+u_\tau\frac{D\boldsymbol{\tau}}{Dt}+\frac{DV}{Dt}\boldsymbol{n},$$

and therefore its projection on the the normal direction is

$$\frac{D\boldsymbol{u}}{Dt}\cdot\boldsymbol{n}=u_\tau\frac{D\boldsymbol{\tau}}{Dt}\cdot\boldsymbol{n}+\frac{DV}{Dt}.$$

Using the equation of motion (3.9) yields

$$\frac{\partial p}{\partial n}=-\rho\frac{D\boldsymbol{u}}{Dt}\cdot\boldsymbol{n}=-\rho u_\tau\frac{D\boldsymbol{\tau}}{Dt}\cdot\boldsymbol{n}-\rho\frac{DV}{Dt}.$$

Furthermore,

$$\frac{D\boldsymbol{\tau}}{Dt}=\frac{\partial\boldsymbol{\tau}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{\tau}=\frac{\partial\boldsymbol{\tau}}{\partial t}+u_\tau\frac{\partial\boldsymbol{\tau}}{\partial\tau}=\frac{\partial\boldsymbol{\tau}}{\partial t}+u_\tau\kappa\boldsymbol{n},$$

which gives

$$\frac{\partial p}{\partial n}=-\rho u_\tau^2\kappa-\rho u_\tau\frac{\partial\boldsymbol{\tau}}{\partial t}\cdot\boldsymbol{n}-\rho\frac{DV}{Dt}. \tag{3.23}$$

Notice that $DV/Dt=\partial V/\partial t+\boldsymbol{u}\cdot\nabla V$ can be easily computed using the analytical expression of V. Finally, one can show that

$$\frac{\partial\boldsymbol{\tau}}{\partial t}\cdot\boldsymbol{n}=-\phi_x\phi_{yt}+\phi_y\phi_{xt},$$

where the subscripts denote the corresponding partial derivatives.

Condition on the density. The condition on the density remains the same as in the fixed boundary case, namely (3.12).

Condition on the tangential component of the velocity. This condition is obtained as in §3.2.1, except that the normal component of the the velocity is no longer zero on the boundary and should be taken into account in (3.13). The boundary condition (3.15) then becomes

$$\frac{\partial u_\tau}{\partial n}=\frac{\partial u_n}{\partial\tau}+u_\tau\kappa,$$

and using (3.21), we obtain

$$\frac{\partial u_\tau}{\partial n}=\frac{\partial V}{\partial\tau}+u_\tau\kappa, \tag{3.24}$$

where $\partial V/\partial\tau$ is assigned from the motion of the obstacle. Again, if one considers the case of a translational motion of a disk of radius R centered at $\boldsymbol{c}(t)\equiv(x_c(t),y_c(t))$ and moving with given velocity $\boldsymbol{u}^{\mathbf{c}}(t)$, then $V=\boldsymbol{n}\cdot\boldsymbol{u}^c(t)$,

$$\frac{\partial V}{\partial\tau}=\frac{\partial\boldsymbol{n}}{\partial\tau}\cdot\boldsymbol{u}^c+\frac{\partial\boldsymbol{u}^c}{\partial\tau}\cdot\boldsymbol{n}=-\kappa\boldsymbol{\tau}\cdot\boldsymbol{u}^c=-u_\tau^c\kappa,$$

and therefore the boundary condition (3.24) becomes

$$\frac{\partial u_\tau}{\partial n}=(\boldsymbol{u}-\boldsymbol{u}^c)_\tau\kappa. \tag{3.25}$$

3.2.4 Discretization of the Moving Boundary Conditions

We now turn to a numerical discretization of the boundary conditions (3.12), (3.21), (3.23) and (3.24), which are valid in the case of moving boundary. Unlike the fixed boundary case, here we will need to compute the values of u, v, p and ρ at the ghost points in both first, $\mathcal{L}_1$, and second, $\mathcal{L}_2$, layers of ghost points.

Let us first consider a ghost point $G \in \mathcal{L}_2$. As in the fixed boundary case, we use formula (3.17) to compute the projection point $B \in \partial\Omega$, which is the closest point to G on the boundary (see Figure 3.5). We then define two 3×3 stencils associated with the ghost point G: $\mathcal{S}_G^I$ and $\mathcal{S}_G^{II}$ also outlined in Figure 3.5. Note that both stencils consists of ghost and interior points only and that the boundary point B is contained in both $\mathcal{S}_G^I$ and $\mathcal{S}_G^{II}$.

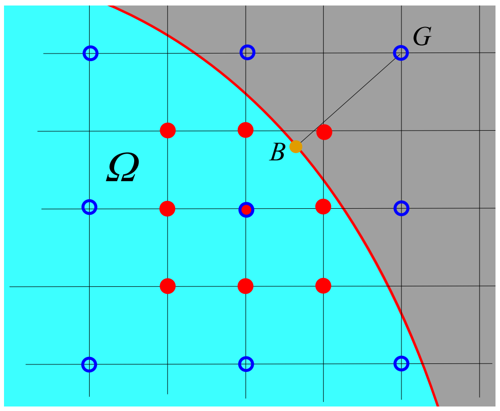

Figure 3.5: Ghost point $G \in \mathcal{L}_2$, its boundary projection $B \in \partial\Omega$, and two stencils: $\mathcal{S}_G^I$ (red filled circular points) and $\mathcal{S}_G^{II}$ (blue unfilled circular points).

Then, the discretization of the boundary conditions is obtained by using the corresponding bi-quadratic approximation in (3.12), (3.21), (3.23) and (3.24):

$$
\begin{aligned}
&\mathcal{Q}[u_n;\mathcal{S}_G^{II}](B) = V(B),\\
&\frac{\partial \mathcal{Q}[u_\tau;\mathcal{S}_G^{II}](B)}{\partial n} = \kappa \mathcal{Q}[u_\tau;\mathcal{S}_G^{II}](B),\\
&\frac{\partial \mathcal{Q}[p;\mathcal{S}_G^{II}](B)}{\partial n} = -\mathcal{Q}[\rho;\mathcal{S}_G^{I}](B)\Big\{\kappa\big(\mathcal{Q}[u_\tau;\mathcal{S}_G^{I}](B)\big)^2\\
&\qquad\qquad - \mathcal{Q}[u_\tau;\mathcal{S}_G^{I}](B)\left[\frac{\partial\phi(B)}{\partial x}\frac{\partial^2\phi(B)}{\partial y\partial t} + \frac{\partial\phi(B)}{\partial y}\frac{\partial^2\phi(B)}{\partial x\partial t}\right]\\
&\qquad\qquad + \frac{\partial V(B)}{\partial t} + \mathcal{Q}[u;\mathcal{S}_G^{I}](B)\frac{\partial V(B)}{\partial x} + \mathcal{Q}[v;\mathcal{S}_G^{I}](B)\frac{\partial V(B)}{\partial y}\Big\},\\
&\frac{\partial \mathcal{Q}[\rho;\mathcal{S}_G^{II}](B)}{\partial n} = \frac{\mathcal{Q}[\rho;\mathcal{S}_G^{II}](B)}{\gamma\mathcal{Q}[p;\mathcal{S}_G^{I}](B)}\cdot\frac{\partial \mathcal{Q}[p;\mathcal{S}_G^{I}](B)}{\partial n},
\end{aligned}
\tag{3.26}
$$

where, as before, $\mathcal{Q}[\psi;\mathcal{S}]$ is a bi-quadratic interpolant of a grid function $\{\psi_{j,k}\}$ in the stencil $\mathcal{S}$. Notice that in each of the four equations in (3.26) the quantity on the left-hand side and the same quantities on the right-hand side (RHS) are discretized using the larger stencil $\mathcal{S}_G^{II}$ (this is needed to include the sought values of $u(G)$, $v(G)$, $p(G)$ and $\rho(G)$ into the system), while the other quantities on the RHS are discretized in a more accurate manner using the smaller stencil $\mathcal{S}_G^I$.

Finally, we follow the approach described in §3.2.2 and transform the system of boundary conditions (3.12), (3.21), (3.23), (3.24) into a time-dependent problem with a fictitious time and compute its steady-state solution using an iterative scheme similar to the one presented in §3.2.2. Observe that Remark 3.5 for the fixed boundary case holds also in the case moving boundaries.

Remark 3.6 Note that if the ghost point G belongs to the first layer of ghost points, that is, if $G \in \mathcal{L}_1$, the boundary procedure presented in this section is substantially simplified since in this case $\mathcal{S}_G^{II} \equiv \mathcal{S}_G^I$ and only one interpolant $\mathcal{Q}[\,\cdot\,;\mathcal{S}_G^I]$ is used for each interpolated quantity.

4 Numerical Examples

In this section, we perform several numerical tests and demonstrate the performance of the proposed numerical method (a special focus will be put on the accuracy of the numerical boundary treatment).

In all of the examples, we take the specific heat ratio constant $\gamma = 1.4$ and set the minmod parameter in (2.7) and (3.6) to be $\theta = 1.5$.

4.1 One-Dimensional Example

We begin with a 1-D example, in which we numerically study a gas in a tube with a moving right boundary located at $x = x_B(t)$.

Example 1—One-Dimensional Accuracy Test

In this example, we take the computational domain to be $[0, 1]$ with the solid wall boundary conditions imposed at both ends. While the left boundary is fixed, we assume that the right boundary oscillates according to the a-priori prescribed equation of motion

$$x_B(0) = 0.9, \quad x_B'(t) = 0.25\sin^3(2\pi t).$$

The initial data are taken to be constant:

$$\rho(x, 0) \equiv 1, \quad u(x, 0) \equiv 0, \quad p(x, 0) \equiv 1.$$

In Figure 4.1, we show the density computed by the FD scheme (2.3)–(2.9) on a uniform spatial grid with $\Delta x = 1/200$ at times $t = 0.25$, 0.5, 0.75 and 1.00. As one can clearly see, the solution at times $t = 0.25$, 0.5 and 0.75 is smooth, while by the time $t = 1$ the solution breaks down and a shock wave develops.

In order to demonstrate the second order of accuracy, we study the self-convergence of the computed solutions. To this end, we compute the solutions on 5 uniform grids with $\Delta x = 1/200$, 1/400, 1/800, 1/1600 and 1/3200 and measure the L^1- and L^∞-norms of the differences between the solutions computed on two consecutive grids. To compute the experimental rates of convergence,

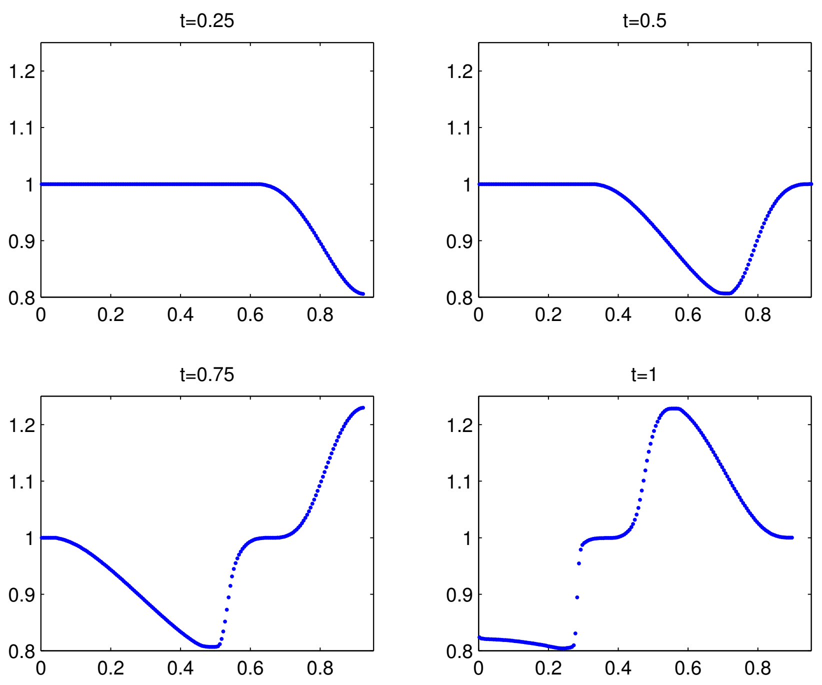

Figure 4.1: Example 1: Density computed using the proposed FD scheme at four different times.

r_1 and r_∞, at the uniform grid of size Δx, we take the solutions on two coarser grids of size $2\Delta x$ and $4\Delta x$ and use Aitken's formula [1] to obtain

$$r = \log_2\left(\frac{\|\rho_{2\Delta x} - \rho_{4\Delta x}\|}{\|\rho_{\Delta x} - \rho_{2\Delta x}\|}\right).$$

The obtained errors and experimental rates of convergence are shown in Table 4.1. We note that the global L^1 convergence rates of the proposed method are about 2 as expected, while the L^∞ convergence rates are slightly lower. The latter can be explained by the clipping phenomenon typical for the generalized minmod reconstruction.

Δx	$\|\rho_{\Delta x} - \rho_{2\Delta x}\|_1$	r_1	$\|\rho_{\Delta x} - \rho_{2\Delta x}\|_\infty$	r_∞
1/400	$2.05 \cdot 10^{-4}$	–	$3.90 \cdot 10^{-3}$	–
1/800	$5.03 \cdot 10^{-5}$	2.03	$1.31 \cdot 10^{-3}$	1.58
1/1600	$1.29 \cdot 10^{-5}$	1.96	$3.82 \cdot 10^{-4}$	1.77
1/3200	$3.25 \cdot 10^{-6}$	1.99	$1.04 \cdot 10^{-4}$	1.88

Table 4.1: Example 1: L^1- and L^∞-errors and the experimental convergence rates at time $t = 0.75$.

4.2 Two-Dimensional Examples

In the 2-D examples, the computational domain is $[0,1]\times[0,1]$ and the fluid domain is $\Omega = [0,1]\times[0,1]\setminus\mathcal{B}_R(t)$, where $\mathcal{B}_R(t)$ is the rigid ball of radius $R=0.1$ and center $(x_c(t),y_c(t))$, see Figure 4.2. We implement inflow boundary conditions at the left boundary and outflow boundary conditions at the right boundary, while the top and the bottom boundaries of the domain as well as the boundaries of the moving circle are assumed to be solid walls.

Instead of adopting the boundary treatment proposed in §3.2, a simpler (but less accurate) approach can be obtained by an extrapolation of p, ρ and u_τ from inside the domain to the ghost points without exploiting the physical meaning of these quantities on the boundary. The simplest way to extrapolate these values is by using a constant extrapolation along the normal direction to the boundary, or, alternatively, prescribing vanishing Neumann conditions on the boundary. In this case, the set of boundary conditions will read as

$$\frac{\partial\rho}{\partial n}=0,\quad u_n=V,\quad \frac{\partial u_\tau}{\partial n}=0,\quad \frac{\partial p}{\partial n}=0. \tag{4.1}$$

We will compare the numerical solutions obtained using two different boundary conditions implemented at the surface of the ball: (i) the boundary conditions proposed in §3.2 and (ii) the constant extrapolation (4.1).

To better illustrate the advantages of the proposed boundary treatment, we will plot the 1-D slices of the computed solutions along the three directions outlined in Figure 4.2.

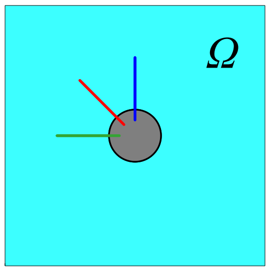

Figure 4.2: 2-D examples: Computational domain and vertical (blue), oblique (red) and horizontal (green) directions, along which the 1-D slices of the computed solutions will be presented.

Example 2—Two-Dimensional Accuracy Test

In the first 2-D example, we consider a simple wave that propagates around a steady disk with $x_c(t)\equiv 0.6$ and $y_c(t)\equiv 0.5$ in a constant medium. The initial conditions are

$$(\rho(x,y,0),u(x,y,0),v(x,y,0),p(x,y,0))=\begin{cases}\big(\widetilde{\rho}(x,y),\widetilde{u}(x,y),0,\widetilde{p}(x,y)\big), & \text{if } |x-0.35|<0.25,\\ \big(\rho_0,0,0,p_0\big), & \text{otherwise,}\end{cases}$$

where

$$\widetilde{u}(x,y)=0.5\,e^{-\frac{(x-0.35)^2}{0.005}},\quad \widetilde{\rho}(x,y)=\rho_0\left(1+\frac{\gamma-1}{2}\cdot\frac{\widetilde{u}(x,y)}{c_0}\right)^{\frac{2}{\gamma-1}},\quad \widetilde{p}(x,y)=p_0\left(\frac{\widetilde{\rho}(x,y)}{\rho_0}\right)^{\gamma}.$$

We set $\rho_0=p_0=1$ and $c_0=\sqrt{\gamma p_0/\rho_0}=\sqrt{1.4}$.

Note that these initial data indeed correspond to a simple wave since in the absence of the obstacle, the exact solution would be

$$u(x,y,t)=u(\xi,y,0),\quad \rho(x,y,t)=\rho_0\left(1+\frac{\gamma-1}{2}\cdot\frac{u(x,y,t)}{c_0}\right)^{\frac{2}{\gamma-1}},\quad p(x,y,t)=p_0\left(\frac{\rho(x,y,t)}{\rho_0}\right)^{\gamma},$$

where ξ satisfies

$$x=\xi+\Big(c_0+\frac{\gamma+1}{2}u(\xi,y,0)\Big)t.$$

We performed the simulation until the final time $t=0.2$ on four different uniform grids with $\Delta x=\Delta y=1/50$, $1/100$, $1/200$ and $1/400$. The density $\rho(x,y,0.2)$ computed on the finest of these grids is shown in Figure 4.3 together with the initial condition $\rho(x,y,0)$.

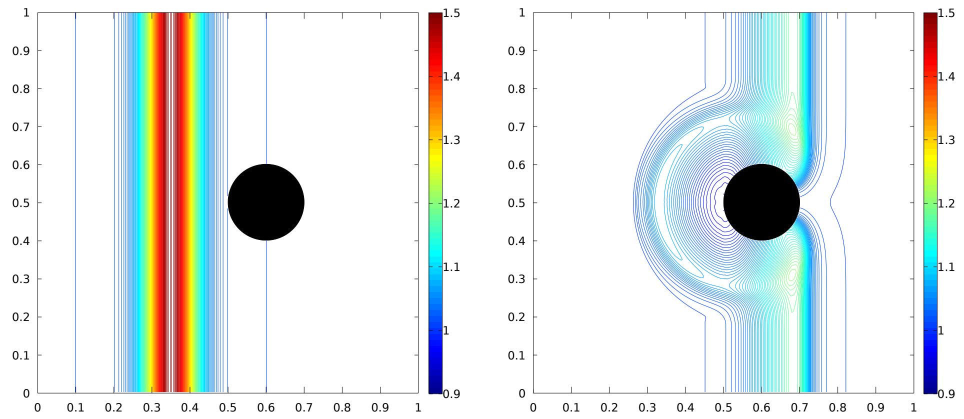

Figure 4.3: Example 2: Density computed by the proposed FD scheme at times $t=0$ (left) and $t=0.2$ (right).

At time $t=0.2$, the exact solution is still smooth, but due to the interaction with the disk its analytic expression is unavailable. We therefore once again use Aitken's formula to estimate the L^1-errors and compute the experimental convergence rates, which are presented in Table 4.2 for the density ρ and x-component of the velocity u. As one can see, the convergence rates approach 2 for finer grids.

$\Delta x=\Delta y$	$\Vert\rho_{\Delta x}-\rho_{2\Delta x}\Vert_1$	rate	$\Vert u_{\Delta x}-u_{2\Delta x}\Vert_1$	rate
1/100	$2.71\cdot10^{-3}$	–	$2.62\cdot10^{-3}$	–
1/200	$1.09\cdot10^{-3}$	1.31	$1.08\cdot10^{-3}$	1.28
1/400	$3.12\cdot10^{-4}$	1.80	$3.11\cdot10^{-4}$	1.80

Table 4.2: Example 2: L^1-errors for ρ and u and the corresponding experimental convergence rates.

Finally, we compare the results obtained by the same second-order FD scheme but with two different boundary conditions: equations (3.8), (3.15), (3.11) and (3.12) and the constant extrapolation (4.1). In Figure 4.4, we show the 1-D slices of the corresponding solutions along the blue line, that is, in the vertical direction of Figure 4.2. As one can clearly see, the use of the proposed boundary conditions leads to a substantial improvement of the computed solution near the boundary. This is especially pronounced in the tangential velocity field.

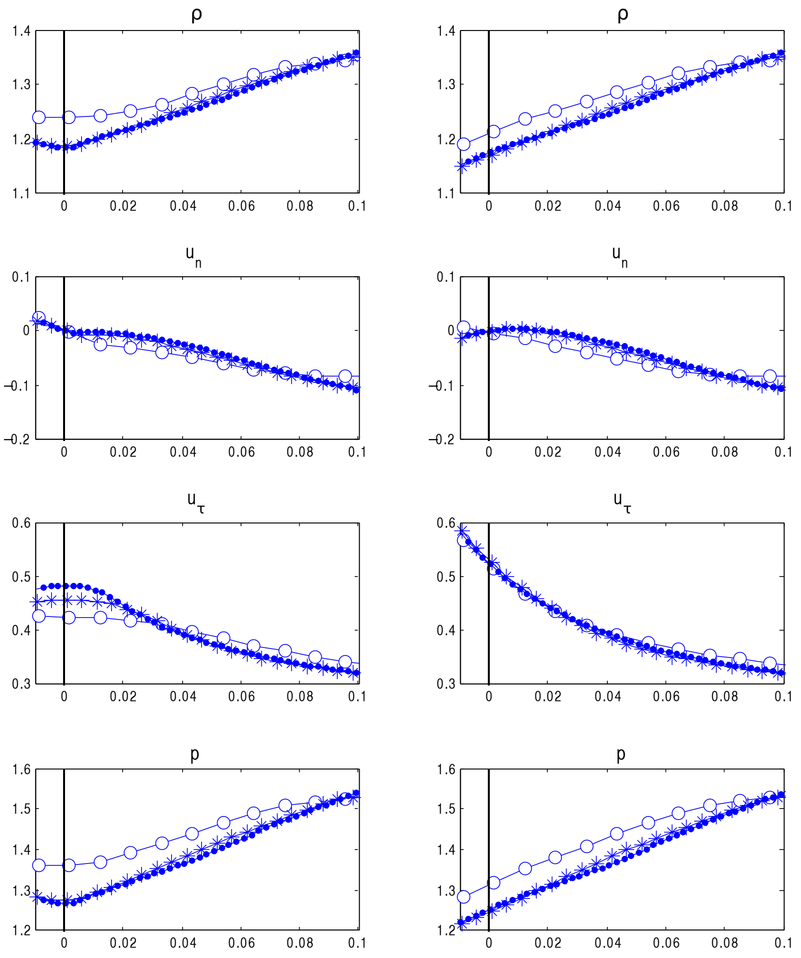

Figure 4.4: Example 2: Density, normal and tangential velocities and pressure as functions of the signed distance from the boundary in the vertical direction at $t = 0.2$. The results are computed by the proposed FD scheme with $\Delta x = \Delta y = 1/50$ (circles), $1/100$ (stars) and $1/200$ (dots) using the boundary conditions (3.8), (3.15), (3.11) and (3.12) (right) and the constant extrapolation (4.1) (left).

Example 3—Shock Hitting a Steady Disk

In this example, we perform the moving shock-steady disk test from [8, Example 3]. We consider a flow generated by a vertical shock which moves from its initial position at $x = 0.25$ to the right, hitting a steady disk with $x_c(t) \equiv y_c(t) \equiv 0.5$. The initial conditions are:

$$(\rho(x,y,0), u(x,y,0), v(x,y,0), p(x,y,0)) = \begin{cases} (4/3, 35/99, 0, 1.5), & x \le 0.25, \\ (1, 0, 0, 1), & x > 0.25. \end{cases}$$

We perform the simulation until the final time $t = 0.4$ on three different uniform grids with $\Delta x = \Delta y = 1/50$, $1/100$ and $1/200$. We compare the results obtained by the same second-order FD scheme but with two different boundary conditions: the boundary conditions (3.8), (3.15), (3.11), and (3.12) and the constant extrapolation (4.1). In Figures 4.5 and 4.6, we show the 1-D slices of the corresponding solutions along the vertical (blue) and oblique (red) lines, respectively, of Figure 4.2. As one can clearly see, the use of the proposed boundary condition drastically improves the quality of the results near the boundary.

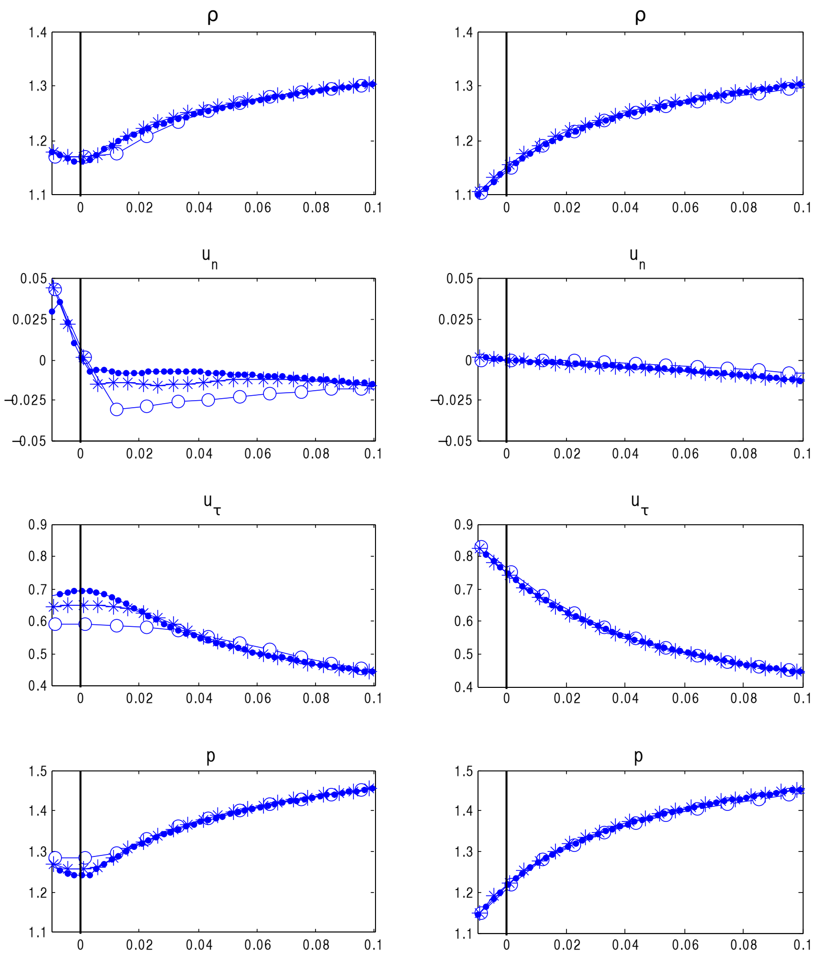

Figure 4.5: Example 3: Density, normal and tangential velocities and pressure as functions of the signed distance from the boundary in the vertical direction at $t = 0.4$. The results are computed by the proposed FD scheme with $\Delta x = \Delta y = 1/50$ (circles), $1/100$ (stars) and $1/200$ (dots) using the boundary conditions (3.8), (3.15), (3.11) and (3.12) (right) and the constant extrapolation (4.1) (left).

Example 4—Moving Ball

In the final example taken from [8, Example 4], we perform a moving boundary simulation. A ball oscillates up and down in a translational motion. The center of the ball is initially located at $x_c(0) = y_c(0) = 0.5$ and moves with velocity $(x_c'(t), y_c'(t)) = (0, 0.1\pi\cos(10\pi t))$.

We perform the simulation until the final time $t = 0.2$ (after one period of the motion of the ball center) on three different uniform grids with $\Delta x = \Delta y = 1/50$, $1/100$ and $1/200$. We compare the results obtained by the same second-order FD scheme but with two different boundary conditions: the boundary conditions (3.21), (3.24), (3.23) and (3.12) and the constant extrapolation (4.1). In Figures 4.7, 4.8 and 4.9, we show the 1-D slices of the corresponding solutions along the vertical (blue), oblique (red) and horizontal (green) lines, respectively, of Figure 4.2. Also in this case there is a noticeable improvement in the numerical solution when adopting the proposed boundary conditions, even in the vertical slice, for which obviously $u_\tau = 0$.

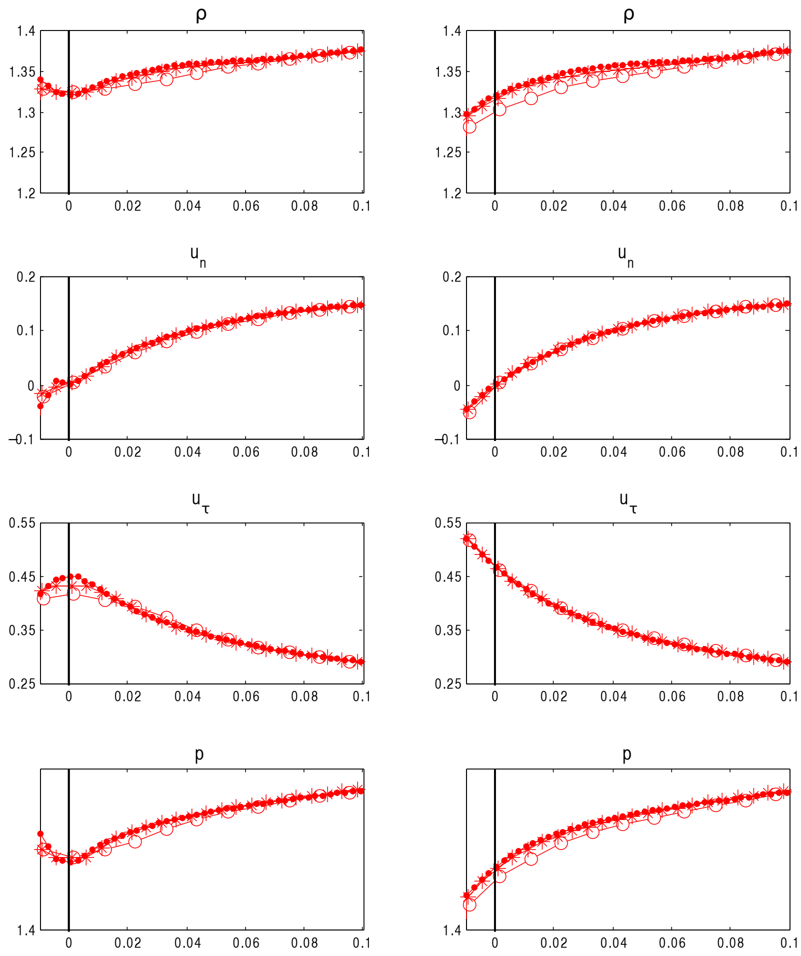

Figure 4.6: Example 3: Density, normal and tangential velocities and pressure as functions of the signed distance from the boundary in the oblique direction at $t = 0.4$. The results are computed by the proposed FD scheme with $\Delta x = \Delta y = 1/50$ (circles), $1/100$ (stars) and $1/200$ (dots) using the boundary conditions (3.8), (3.15), (3.11) and (3.12) (right) and the constant extrapolation (4.1) (left).

5 Conclusions

In this paper, we presented a new FD shock-capturing scheme for the numerical solution of the Euler equations of gas dynamics on a Cartesian mesh in an arbitrary 2-D domain in the presence of moving boundary. We restricted our consideration to the case in which the changes in the domain boundary are prescribed and occur due to the movement of solid objects/obstacles/walls, which is assumed to be independent of the fluid flow. The core of the proposed method is the following: the computational domain $\Omega(t)$ is embedded in a rectangular region Ω_R, which is discretized by a regular Cartesian grid. Three sets of nodes are defined at each time step: internal nodes (belonging to $\Omega(t)$), *ghost* nodes (external nodes within one or few grid points from an internal node), and inactive nodes (neither internal nor ghost nodes). A conservative FD scheme is used as a space discretization. The evolution of the system in the time interval $[t^n, t^{n+1}]$ is performed as follows: for points that will be internal at time t^{n+1}, the field variables are evolved by integrating the governing equations in time, while boundary conditions are used to compute the values of all the ghost points which are required to close the system of equations. The approach is general

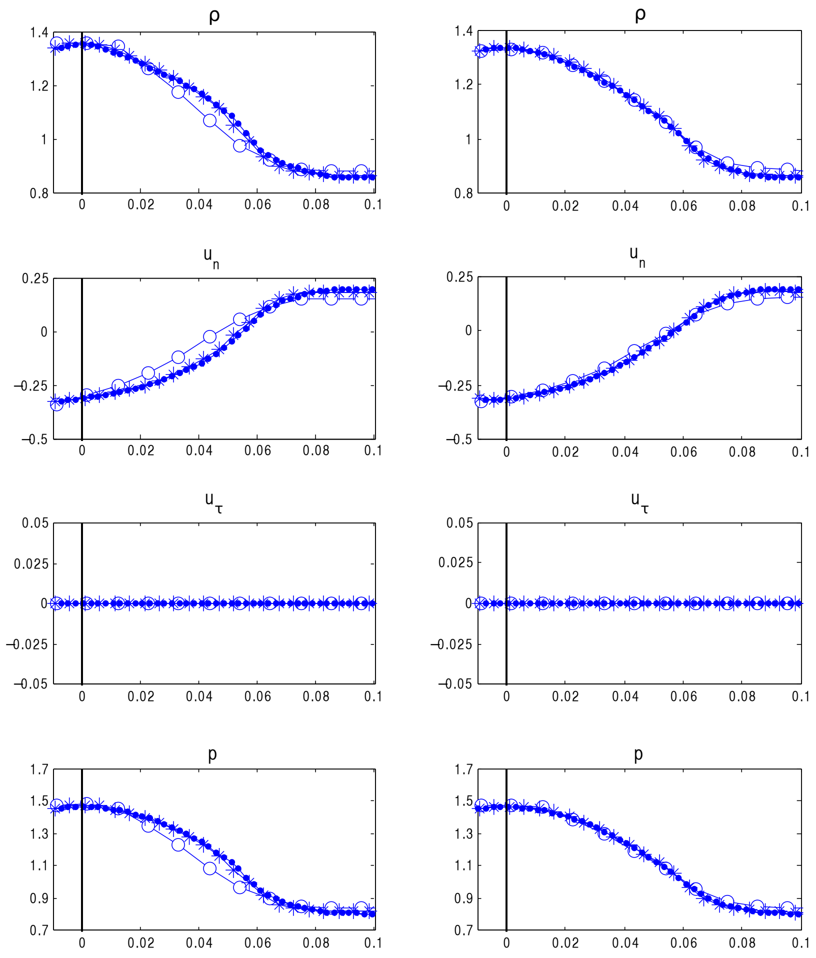

Figure 4.7: Example 4: Density, normal and tangential velocities and pressure as functions of the signed distance from the boundary in the vertical direction at $t = 0.2$. The results are computed by the proposed FD scheme with $\Delta x = \Delta y = 1/50$ (circles), $1/100$ (stars) and $1/200$ (dots) using the boundary conditions (3.21), (3.24), (3.23) and (3.12) (right) and the constant extrapolation (4.1) (left).

and allows a treatment of boundary condition with arbitrary order of accuracy, depending on the accuracy of the interpolation scheme. Here, we limit ourselves to the second order of accuracy only.

Slip wall boundary conditions are considered, that is, the normal velocity of the fluid is equal to the normal velocity of the boundary, while no restrictions are imposed on the tangential component. A detailed description of proper boundary conditions for Euler equations, motivated by physical principles, is provided; such boundary conditions allow the assignment of all four components of the field vector (density, pressure and the two components of the velocity) at each ghost point.

The effectiveness of the proposed technique is checked *a-posteriori* by detailed numerical simulations, both in one and two space dimensions, and both for fixed and moving boundaries. The expected second-order accuracy has been numerically verified both in the 1-D and 2-D cases. A boundary condition is derived for the normal derivative of the tangential component of the fluid velocity. The validity of such condition is verified by comparing the numerical results with

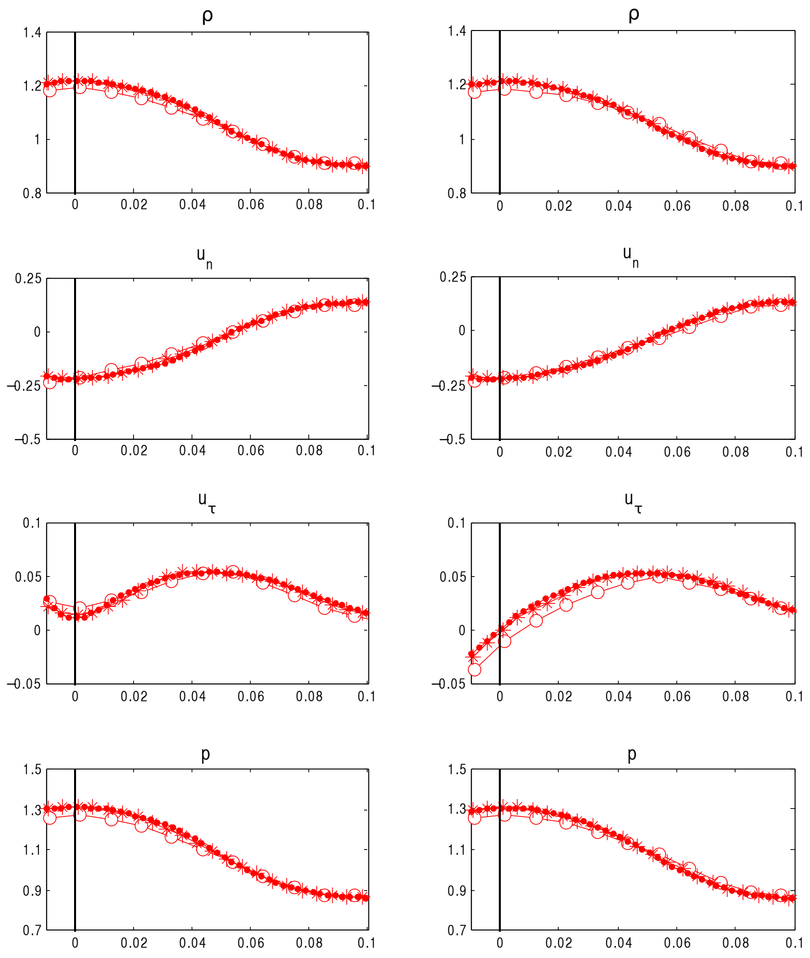

Figure 4.8: Example 4: Density, normal and tangential velocities and pressure as functions of the signed distance from the boundary in the oblique direction at $t = 0.2$. The results are computed by the proposed FD scheme with $\Delta x = \Delta y = 1/50$ (circles), $1/100$ (stars) and $1/200$ (dots) using the boundary conditions (3.21), (3.24), (3.23) and (3.12) (right) and the constant extrapolation (4.1) (left).

those obtained when imposing the trivial first-order boundary condition that assumes zero normal derivative of the tangential component of the velocity. Much better results are obtained when the proper boundary condition is adopted for the normal derivative of the tangential component of the velocity.

Although in the present paper we assume that the (possibly moving) geometry of the domain is known *a-priori*, the approach is general and can be extended to the case in which the geometry itself is an unknown of the problem. The coupling between the fluid and one or several rigid bodies suspended in it can be treated as described in §5.1 below. For more complex objects, for example for deformable ones, the boundary of the domain Ω would satisfy a suitable level set equation, which is coupled with the evolution equation of the gas. This topic is a subject of future study.

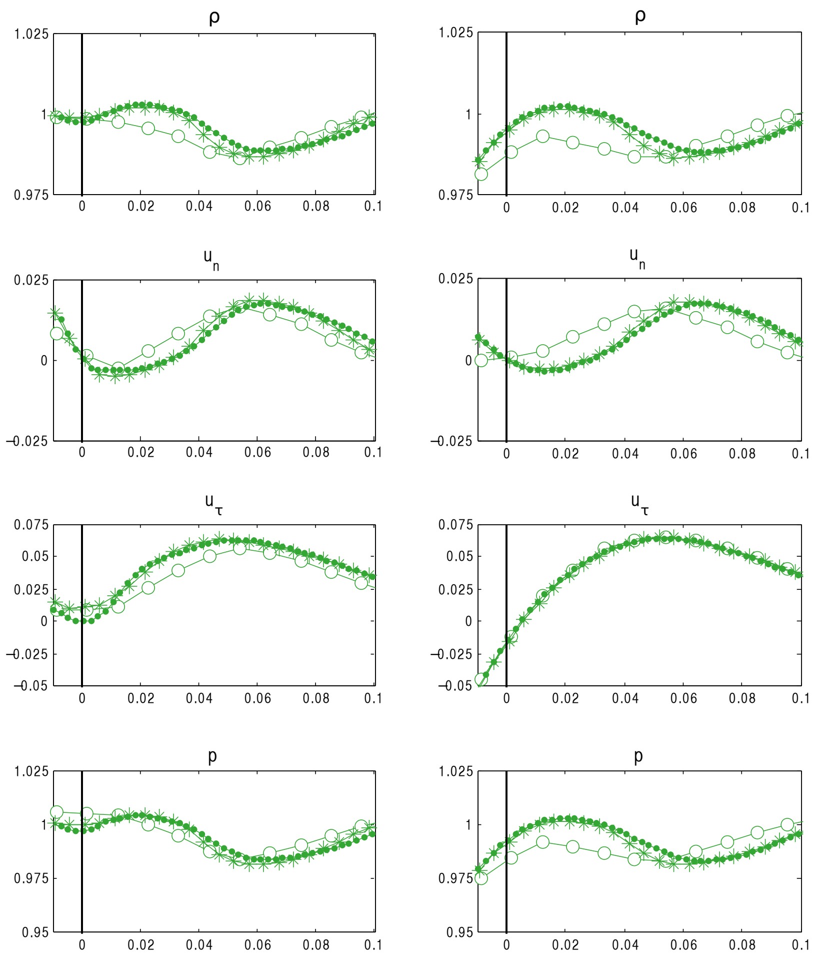

Figure 4.9: Example 4: Density, normal and tangential velocities and pressure as functions of the signed distance from the boundary in the oblique direction at $t = 0.2$. The results are computed by the proposed FD scheme with $\Delta x = \Delta y = 1/50$ (circles), $1/100$ (stars) and $1/200$ (dots) using the boundary conditions (3.21), (3.24), (3.23) and (3.12) (right) and the constant extrapolation (4.1) (left).

5.1 Suspended objects

The presented method can be used to describe the interaction between the gas and a suspended solid object, which moves as a result of the actions that the gas exerts on it.

Let us denote by $D(t)$ the region occupied by a rigid object and let M be its mass (per unit thickness). Let us also denote by $\boldsymbol{x}_D(t) := (x_D(t), y_D(t))$ the center of mass of the object and by I_D its barycentric moment of inertia. If the object is homogeneous and its density is ρ_D, then

$$M = \rho_D |D|, \quad I_D = \rho_D \int_D |\boldsymbol{x} - \boldsymbol{x}_D|^2 \, d\boldsymbol{x},$$

where by $|D|$ we denote the area of D.

The object D has three degrees of freedom, for example, the coordinates of the center of mass, $\boldsymbol{x}_D$ and the angle θ that the x-axis of a frame of references moving with the body forms with the

x-axis of the fixed frame of reference. The motion of the object can be described by the equations for the dynamics of a rigid body:

$$\dot{\boldsymbol{x}}_G = \boldsymbol{v}_D, \quad M\,\dot{\boldsymbol{v}}_D = \boldsymbol{F}, \quad \dot{\theta} = \omega, \quad I_D\,\dot{\omega} = T_z,$$

where $\boldsymbol{F}$ and T_z denote the total force (per unit thickness) and torque (per unit thickness), respectively, that the gas exerts on the object D. The total force can be obtained from the pressure as

$$\boldsymbol{F} = -\int\limits_{\Gamma} p(\boldsymbol{x})\,\boldsymbol{n}(\boldsymbol{x})\,d\Gamma,$$

where $\Gamma := \partial D$ denotes the boundary of D, and we used the convention that the unit normal $\boldsymbol{n}$ to Γ points out of D (and therefore points *inside* Ω), as illustrated in Figure 5.1 (right). The total torque acting on D can be computed as

$$\boldsymbol{T} = -\int\limits_{\Gamma} (\boldsymbol{x} - \boldsymbol{x}_D) \times \boldsymbol{n}(\boldsymbol{x}) p(\boldsymbol{x})\,d\Gamma.$$

In practice, only the z-component of the torque, T_z, will be nonzero. Such component can be written as

$$T_z = \int\limits_{\Gamma} (\boldsymbol{x} - \boldsymbol{x}_D) \cdot \boldsymbol{\tau}(\boldsymbol{x}) p(\boldsymbol{x})\,d\Gamma.$$

Both the total force $\boldsymbol{F}$ and torque T_z can be easily computed to second order accuracy using the information that is already available on the node and ghost points. Let us denote by q the generic grid square with four grid points at its corners, and let us denote by $\mathcal{Q}$ the set of such grid squares that intersect Γ. Generically, each intersection identifies a segment $\Delta\Gamma$, whose endpoints will be denoted by $\boldsymbol{x}_q^+$ and $\boldsymbol{x}_q^-$; see Figure 5.1 (right). The value of the pressure at each endpoint can then be computed by the 1-D linear interpolation between the values at an internal and ghost nodes.

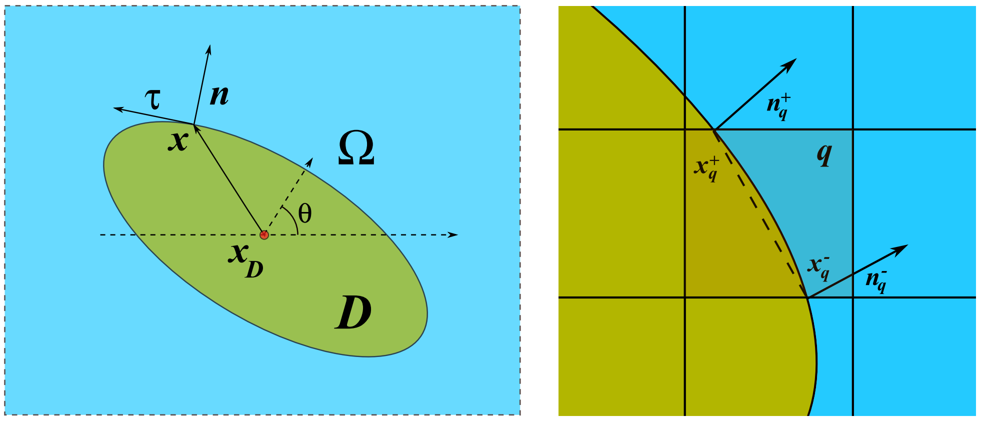

Figure 5.1: Setting for the computation of the total force and torque, obtained by integrating along the boundary of the object D (left); the square q, segment $\Delta\Gamma_q$ (dashed line) and unit normals at the end points used for the computation of a contribution to the total force F and torque T (right).

Then, using the trapezoidal rule on each of such segments, we obtain

$$\boldsymbol{F} = -\frac{1}{2}\sum_{q\in\mathcal{Q}} |\Delta\Gamma_q| \left(p(\boldsymbol{x}_q^+)\boldsymbol{n}(\boldsymbol{x}_q^+) + p(\boldsymbol{x}_q^-)\boldsymbol{n}(\boldsymbol{x}_q^-)\right) + \mathcal{O}(h^2),$$

where $|\Delta\Gamma_q|$ denotes the length of the segment $\Delta\Gamma_q$ and $h := \max(\Delta x, \Delta y)$. A similar expression can be obtained for the torque:

$$T_z = \frac{1}{2}\sum_{q\in\mathcal{Q}} |\Delta\Gamma_q| \left(p(\boldsymbol{x}_q^+)\boldsymbol{\tau}(\boldsymbol{x}_q^+)\cdot(\boldsymbol{x}_q^+ - \boldsymbol{x}_D) + p(\boldsymbol{x}_q^-)\boldsymbol{\tau}(\boldsymbol{x}_q^-)\cdot(\boldsymbol{x}_q^- - \boldsymbol{x}_D)\right) + \mathcal{O}(h^2). \tag{5.1}$$

The treatment of the interaction with a rigid body then just requires to consider six scalar ODEs in addition to the system (3.2). Since only the points near the boundary are involved, the computation of $\boldsymbol{F}$ and T_z leads to only a small increase in the computational time compared to the case in which the motion of the object is prescribed. In the case of a circular disk, the torque is zero since $\boldsymbol{x} - \boldsymbol{x}_D$ and $\boldsymbol{\tau}$ are orthogonal, and therefore ω is constant and has no influence on the dynamics.

If more than one object is immersed in the gas, the interaction between the gas and immersed objects can be similarly treated by solving the equations of the dynamics of the various objects until the objects touch. When this occurs, suitable collision rules should be adopted.

Application of the above formulae (supplemented by collision rules) for studying the motion of rigid bodies in an inviscid compressible flow is the subject of ongoing research.

Acknowledgment. The work of A. Chertock was supported in part by the NSF Grants DMS-1216974 and DMS-1521051. The work of A. Kurganov was supported in part by the NSF Grants DMS-1216957 and DMS-1521009. The work of G. Russo was supported partially by the University of Catania, project F.I.R. *Charge transport in graphene and low dimensional systems*, and partially by ITN-ETN Horizon 2020 Project *ModCompShock, Modeling and Computation on Shocks and Interfaces*, Project Reference 642768.

References

[1] K. E. Atkinson, *An introduction to numerical analysis*, John Wiley & Sons Inc., New York, second ed., 1989.

[2] A. Baeza, P. Mulet, and D. Zorío, *High order boundary extrapolation technique for finite difference methods on complex domains with Cartesian meshes*, J. Sci. Comput., 66 (2016), pp. 761–791.

[3] ——, *High order weighted extrapolation for boundary conditions for finite difference methods on complex domains with Cartesian meshes*, J. Sci. Comput., 69 (2016), pp. 170–200.

[4] F. Bassi and S. Rebay, *High-order accurate discontinuous finite element solution of the 2D Euler equations*, J. Comput. Phys., 138 (1997), pp. 251–285.

[5] A. Chaudhuri, A. Hadjadj, and A. Chinnayya, *On the use of immersed boundary methods for shock/obstacle interactions*, J. Comput. Phys., 230 (2011), pp. 1731–1748.

[6] A. Chaudhuri, A. Hadjadj, O. Sadot, and E. Glazer, *Computational study of shock-wave interaction with solid obstacles using immersed boundary methods*, Internat. J. Numer. Methods Engrg., 89 (2012), pp. 975–990.

[7] A. Chertock, S. Karni, and A. Kurganov, *Interface tracking method for compressible multifluids*, M2AN Math. Model. Numer. Anal., 42 (2008), pp. 991–1019.

[8] A. Chertock and A. Kurganov, *A simple Eulerian finite-volume method for compressible fluids in domains with moving boundaries*, Commun. Math. Sci., 6 (2008), pp. 531–556.

[9] A. Coco, G. Currenti, C. Del Negro, and G. Russo, *A second order finite-difference ghost-point method for elasticity problems on unbounded domains with applications to volcanology*, Commun. Comput. Phys., 16 (2014), pp. 983–1009.

[10] A. Coco and G. Russo, *Finite-difference ghost-point multigrid methods on Cartesian grids for elliptic problems in arbitrary domains*, J. Comput. Phys., 241 (2013), pp. 464–501.

[11] A. Dadone, *Symmetry techniques for the numerical solution of the 2D Euler equations at impermeable boundaries*, Internat. J. Numer. Methods Fluids, 28 (1998), pp. 1093–1108.

[12] A. Dadone and B. Grossman, *Surface boundary conditions for the numerical solution of the Euler equations*, AIAA Journal, 32 (1994), pp. 285–293.

[13] ——, *Ghost-cell method for inviscid two-dimensional flows on Cartesian grids*, AIAA Journal, 42 (2004), pp. 2499–2507.

[14] ——, *Ghost-cell method for analysis of inviscid three-dimensional flows on Cartesian-grids*, Comput. & Fluids, 36 (2007), pp. 1513–1528.

[15] A. du Chéné, C. Min, and F. Gibou, *Second-order accurate computation of curvatures in a level set framework using novel high-order reinitialization schemes*, J. Sci. Comput., 35 (2008), pp. 114–131.

[16] R. Fazio and G. Russo, *Central schemes and second order boundary conditions for 1D interface and piston problems in Lagrangian coordinates*, Commun. Comput. Phys., 8 (2010), pp. 797–822.

[17] F. Gibou, R. P. Fedkiw, L.-T. Cheng, and M. Kang, *A second-order-accurate symmetric discretization of the Poisson equation on irregular domains*, J. Comput. Phys., 176 (2002), pp. 205–227.

[18] Y. Gorsse, A. Iollo, H. Telib, and L. Weynans, *A simple second order Cartesian scheme for compressible Euler flows*, J. Comput. Phys., 231 (2012), pp. 7780–7794.

[19] S. Gottlieb, D. Ketcheson, and C.-W. Shu, *Strong stability preserving Runge-Kutta and multistep time discretizations*, World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2011.

[20] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.

[21] L. Krivodonova and M. Berger, *High-order accurate implementation of solid wall boundary conditions in curved geometries*, J. Comput. Phys., 211 (2006), pp. 492–512.

[22] K.-A. Lie and S. Noelle, *On the artificial compression method for second-order nonoscillatory central difference schemes for systems of conservation laws*, SIAM J. Sci. Comput., 24 (2003), pp. 1157–1174.

[23] L. Monasse, V. Daru, C. Mariotti, S. Piperno, and C. Tenaud, *A conservative coupling algorithm between a compressible flow and a rigid body using an embedded boundary method*, J. Comput. Phys., 231 (2012), pp. 2977–2994.

[24] H. Nessyahu and E. Tadmor, *Nonoscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[25] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*, vol. 153 of Applied Mathematical Sciences, Springer-Verlag, New York, 2003.

[26] G. Russo and P. Smereka, *A remark on computing distance functions*, J. Comput. Phys., 163 (2000), pp. 51–67.

[27] J. A. Sethian, *Level set methods and fast marching methods*, vol. 3 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, second ed., 1999. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.

[28] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in Advanced numerical approximation of nonlinear hyperbolic equations (Cetraro, 1997), vol. 1697 of Lecture Notes in Math., Springer, Berlin, 1998, pp. 325–432.

[29] ——, *High order weighted essentially nonoscillatory schemes for convection dominated problems*, SIAM Rev., 51 (2009), pp. 82–126.

[30] C.-W. Shu and S. Osher, *Efficient implementation of essentially nonoscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.

[31] ——, *Efficient implementation of essentially nonoscillatory shock-capturing schemes. II*, J. Comput. Phys., 83 (1989), pp. 32–78.

[32] P. K. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[33] B. van Leer, *Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.