Design of a high-order front tracking method in 2D

By

MEHDI VAHAB

B.S. (Khaje Nasir Toosi University of Technology, Tehran) 2005
M.S. (Chalmers University of Technology, Gothenburg) 2008

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Engineering - Applied Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Gregory H. Miller, Chair

Hector A. Baldis

Walter M. Harris

Committee in Charge

2010

Contents

Mehdi Vahab
September 2010
Engineering - Applied Science

Design of a high-order front tracking method in 2D

Abstract

This document presents a proposal for a new high-order front tracking method in 2D. A thorough review of existing methods for moving and/or irregular boundaries is presented. From this review, the 1D front-tracking approach of Gatti-Bono emerges as the most promising starting point for a higher-dimensional method. The Gatti-Bono method in 1D is then explained in detail, and a 2D extension is proposed. This extension incorporates a number of ideas from the literature on embedded boundary methods for stationary irregular geometries.

Acknowledgments

I would like to thank my adviser Professor Gregory Miller who introduced me to this field. With his guidance and support, I could prepare this thesis. Also, I would like to thank my parents for their everlasting encouragements and believing in me.

Mehdi Vahab

CHAPTER 1

Prior approches to moving interfaces

1.1. Introduction

Treatment of discontinuities is an important subject in fluid dynamics. Bringing more than one material into a system will add interface handling and stability issues into the problem. A moving boundary is the case that we discuss in this report. This is the general case when two different phase are separated with a boundary that can move freely due to the dynamics of the surrounding phases. Such a system has different applications in different fields. We encounter them in most biological systems such as cell-membrane systems and biofilms. Other examples are bubble modeling and simulation of flame.

In this chapter an overview of the research field and different approaches is presented. The next chapter contains the detailed description of a front-tracking method in 1D to grasp the essentials of modeling of such system while keeping the procedures simple to be easier to understand. A proposal for extending the aforementioned method to 2D is presented in Chapter 3. The last chapter brings the conclusion and future works.

Various methods and algorithms have been developed in the field of moving interfaces. A broad distinction between these methods arises from their mechanical system point of view. The two mainstream viewpoints are Lagrangian and Eulerian. Another distinction between these methods is the numerical schemes they are using, which may be categorized in general as finite element methods (FEM) and finite difference methods (also containing finite volume methods). Also it is possible to classify these methods in two groups of front/interface tracking and front/interface capturing. Front tracking methods treat the interface explicitly and front capturing methods take the front as steep gradients over a short distance. These classifications are not strict and there are different hybrid methods that are benefiting from several approaches. In this chapter, a review of the different methods is presented and possible advantages and disadvantages are discussed.

In the Lagrangian description of a moving fluid, mesh points are defined on material particles and move with them, which results in equations without convective terms. Aligning the interface on the moving grid brings a desirable description of the interface. Difficulties arise when material distortion causes entanglement of the mesh, that may be resolved with remeshing (also called rezoning). On the other hand, in the Eulerian description of a fluid, mesh grids are fixed and material distortions can be handled. Because of the fixed grid meshing numerical methods are more straight forward in areas away from interface. The Eulerian description has convective terms in the formulation which may add complications to the numerical methods. Also, describing the interface on a fixed grid may need complex mapping methods.

The main focus in this document is on the methods that treat hyperbolic systems of conservation laws. A general formulation for such system, in D spatial dimension and with m conserved quantities, is as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla.\vec{\mathbf{F}} = S(\mathbf{U}), \tag{1.1}$$

$$\mathbf{U} = \mathbf{U}(\mathbf{x}, t), \quad \mathbf{x} \in \mathbf{\Omega} \subset \mathbb{R}^D, \tag{1.2}$$

$$\vec{\mathbf{F}} = (\mathbf{F}^1 \dots \mathbf{F}^D) = \vec{\mathbf{F}}(\mathbf{U}), \tag{1.3}$$

$$\mathbf{U},\ \mathbf{F}^d \in \mathbb{R}^m, \tag{1.4}$$

where $\mathbf{\Omega}$ is the problem domain. Considering the two-phase flow problem with a moving boundary, the problem domain is decomposed to two sub-domains, one for each fluid, and the moving boundary that separates them (Figure 1.1).

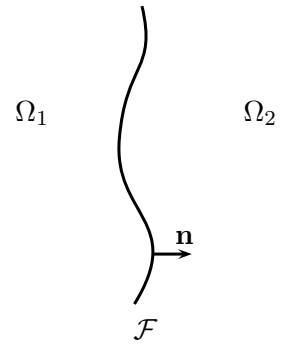

FIGURE 1.1. Two-phase flow domain and the moving interface

1.2. Lagrangian based methods

Early attempts of analysis of moving interface are done within the field of free surface problem. Arbitrary Eulerian-Lagrangian (ALE) methods were developed to overcome the shortcoming of pure Lagrangian or Eulerian methods (see [**19**] and references within for more details on ALE methods). ALE methods were first introduced for finite difference methods and then extended for finite difference methods. The strength of these approach come from the lack of dependency between meshes on which structure, interface and fluid are defined. In other words, the computational grid of fluid may move with an arbitrary velocity with respect to the structural grid. This feature makes it possible to use the appropriate methods for each part of the analysis.

1.2.1. XFEM. One of the issues in the field of moving interfaces is the way to define the interface with respect to computational grids. The same problem came up in the analysis of cracks and fractures and their propagation. The extended finite element method (XFEM) was first introduced by Belytschko and Black [**4**] for modeling arbitrary discontinuities in a function. With a similar methodology, Chessa and Belytschko [**9**] developed a method to model the moving interface in a two-phase flow system. Based on the XFEM method, some enriched shape functions are added to the FEM base functions to have the desired type of discontinuity, e.g. discontinuity in the function or its derivatives. So the solution in the cells which include parts of an interface can approximate the discontinuity at the interface while normal shape functions are applicable to the cells away from the interface (Figure 1.2).

Tracking the interface was done by approximating the interface structure with a level set function. In this method, the level set function is the signed distance to interface and approximated by the same mesh and shape functions as the variables which have the discontinuity on the interface. The standard level set evolution equation, in the form of a variable coefficient advection equation, governs the motion of the interface. For updating the interface position, a cut-off function is also used to reduce the computational contribution from the parts of domain that are far from the interface. Using these structures, the Navier-Stokes equations were solved by a characteristic based split algorithm using projection method of Chorin [**10**]. Examples such as a bubble rising to a free surface and a drop

falling onto a thin film are demonstrated in [**8, 9**] which show the capability of this method for modeling the dramatic topological changes in the interface. Having the fixed mesh makes the analysis easier but a solution for mesh refinement has not been discussed yet.

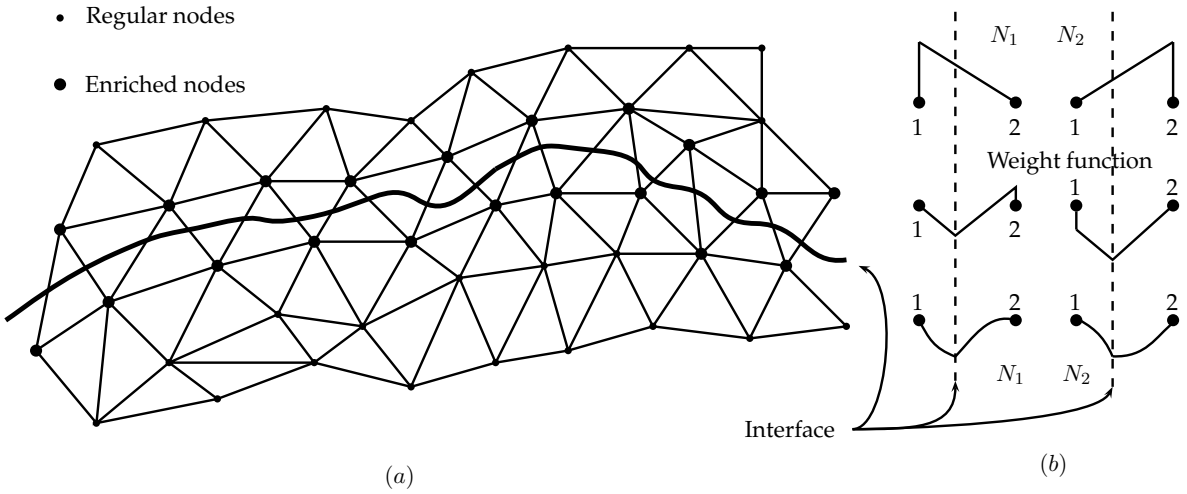

FIGURE 1.2. (a) Finite element mesh with enriched nodes around the interface. (b) Example of finite element shape functions in R^1 .

1.2.2. Cohesive zone method. Another method for modeling arbitrary interfaces is the cohesive zone model introduced by Ortiz and Pandolfi for the crack propagation analysis [**25**]. They use an irregular mesh to represent the domain and the crack propagates on the cell edges. Mesh configuration is re-updated in each step to conform to the crack. Therefore the grid controls the cracks propagation. There are several methods on modeling of fluid-structure interactions that can be extended to models for moving interface. For more on these method using finite element approach see [**18**].

1.3. Eulerian based methods

A vast group of Eulerian based methods are the finite volume methods, defined on Cartesian grids. The interface is expressed by its position on the grid or a volume-of-fluid representation and updated using front tracking/capturing approaches. A short summary of the method introduced by Chern and Colella [**7**] brought here as an example of these methods.

1.3.1. Mass redistribution methods. This method is developed for a two-phase fluid with a moving boundary. Each step starts with updating the position of the front.

The interface divides the domain into two parts and crosses some of the computational cells, making two partitions in those cells. The front speed is needed for finding the position of the front at the next time step. This can be found by solving the Riemann problem on the front. Initial values for the Riemann problem are the average values over space of the variable on each side of the front in the crossed cell (Figure 1.3.a).

Special attention is given to the case when the volume of one of the partitions is very small, because that may lead to an inaccurate average value. To overcome this problem, the average value is calculated not only on the partition of the cell but also on the nearest neighbor cell. The solution to the Riemann problem also provides the state variables on the front which are needed to derive the continuous flux across the front based on the Rankine-Hugoniot relations.

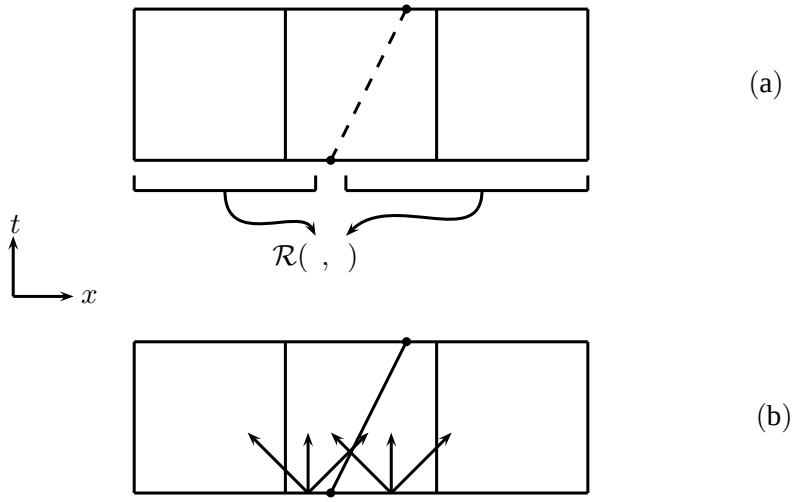

FIGURE 1.3. Chern-Colella method in 1D, (a) Averaging process for solving Riemann problem. (b) Mass redistribution based on characteristics.

To update the solution away from the front an explicit conservative finite difference method is applied that only needs the fluxes on the faces of the cell. These fluxes are accessible through the solution of the Riemann problem. If a computing cell is close to the front, average values of variables on the front, as stated before, should be used instead of the variables from other side of the front.

To update the cells containing the front, the same conservation method may be used while considering that it only applies to a part of the cell. This calculation has a division by the size of the partition which may cause a problem for small partitions. It is possible

to multiply the troublesome flux difference term by the ratio of the partition size to the cell size in order to overcome this instability problem. Scaling the flux which contributes to the update equation makes the method non-conservative. Therefore, the mass difference between the conservative and non-conservative update should be redistributed to the nearby cells to maintain the global conservation. Following the physical intuition, the remaining mass may be propagated according to the characteristics, which is applied by projecting the flux difference on eigenvectors of the system in primitive variables, and modifying the nearby cell values (Figure 1.3.b). Bell et al [**3**] extended the method of Chern and Colella by using the unsplit second-order Godunov algorithm coupled with local adaptive mesh refinement.

1.3.2. *h*-box method. Here, it is noteworthy to mention the h-box method developed by Berger and LeVeque [**5, 6**] for the approximation of hyperbolic conservation laws on irregular grids. They resolved the time step limitation of a small cell by defining a new region, the h-box, with length h equal to the regular cell size, on the edges of the small cell. Variables on an h-box are derived by the weighted average of the values of cells which that h-box covers; that is a local averaging. Then variables and corresponding fluxes on those edges are found by the solution of the Riemann problem which is initialized by the states derived from the h-box cells instead of the original small cell (Figure 1.4). By this procedure the small cell volume problem is fixed. One can use this approach for the moving boundary problem. The only problem is that the geometry of h-boxes are changing while interface moves and one has to recalculate the h-box position in every step, while this is unnecessary in case of fixed irregular cells.

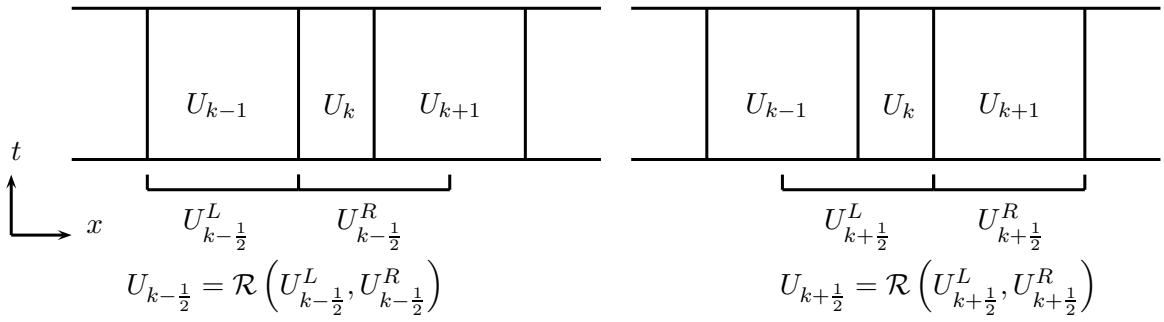

FIGURE 1.4. h-box method to drive the state variables on the edges of an irregular cell.

1.3.3. Effective single phase method. Miller and Puckett developed a numerical method for front capturing in multiple condensed phases [**24**]. They used the volume-of-fluid interface representation. To update the solution on the interface, they used the effective single phase definition for the variables in the cells that contain more than one phase [**13**]. This approach becomes useful in the solution of the approximate Riemann problem on multiphase cell edges. A second-order volume-of-fluid interface reconstruction algorithm was used to update the cells at the front. On a multiphase edge, an edge of a cell which is multiphase or may become multiphase, fluxes for each phase are estimated through a linear approximation to the interface boundary and then finding the contribution of each phase in the upstream direction. The linear approximation is actually a least-squares optimization to the volume-of-fluid representation of the interface. The conservation equations can be decoupled, to some extent, for each phase considering the appropriate constraints on volume fraction of phases and its temporal change in each computational cell. This leads to self-consistent advection equation for volume fraction which can be discretized along the conservation equations for each phase to develop the temporal update procedure. However, it is difficult to apply this method when properties of the adjacent phases are very different, e.s. solid and fluid.

1.3.4. Interface reconstruction method. Miller and Colella developed a method for 3D shock capturing of coupled solid-fluid multiphase systems [**23**]. Similar to [**24**] the dynamics solved for each phase separately in parallel to a volume-of-fluid representation for the interfaces. An extension to the method in [**24**], along with the material properties, all of the state variables are multivalued in a cell that crossed by an interface. This method updates the state variables using a finite volume method for each phase. Therefore, small volume fraction and CFL timestep limitations may happen which are resolved as in [**3**]. The update procedure needs fluxes and variables at the cell edges. A two-step variable update analogous to [**3, 7**] was adopted. Also the mass-weighted redistribution of the mass difference between the two steps of the update is applied to maintain general conservation as in [**7**].

In each step, the interface is reconstructed as the best-fit piecewise planar representation to the volume-of-fluid description of the interface. To advance the fluid fraction, an

advection equation is used for each phase. In this algorithm, the volume of phases advected through faces of the cell is required, which is calculated using a non-specially split advection algorithm (for details see section 3 of [**23**]). It is also notable that for flux calculation, extension by averaging in two passes by the Pilliod scheme was employed, which is similar to the algorithm described in [**3**] for a vector, plus the renormalization at the end of each pass. Away from the interface appropriate single-phase solvers were applied e.g. [**24**].

1.3.5. A second order accurate front tracking method. Gatti-Bono et al developed a front tracking method based on the approaches in [**3, 7, 23**]; a finite volume method with a redistribution algorithm to avoid CFL time-step limitation while maintaining global conservation. In this method, which is second-order accurate away from the front and second-order in the solution in first norm, extra attention was given to the average of the variables on front cells and their difference with the values at the center of the computational cell. These values are different to first-order of the spacial steps.

In this method, the front is explicitly known with its position and speed on the grid. In the procedure for updating with the finite volume method, the front speed is evaluated in two steps. A second-order extrapolation of conserved variables was computed and the corresponding Riemann problem was solved to provide the first approximation of front speed and geometry change. Then variables at the half time step were derived, which gives the fluxes at the half time step. Then, with a non-conservative update one may derive the variables at next time step and consequently the front velocity at next time step. The front velocity at half time step would be the average of this speed and the first approximation, permitting a second-order accurate recalculation of the geometry. A standard Godunov method with the van Leer limiter was employed to find fluxes at time centers and centroids to advance the variables in time using conservative and non-conservative flux differences. Complication may arise in front cells for evaluating the left or right estimate of the primitive variables on an irregular face. This problem is resolved by extrapolation through the front considering the jump condition. After updating variables, the mass difference between the conservative and non-conservative method are redistributed to the neighbor cells with respect to the characteristics while considering reflection and refraction of the waves on the front.

1.4. Hybrid methods

In this context, there are several hybrid methods using both Lagrangian and Eulerian views. Some of these methods are brought here without details of the algorithms just to emphasize the the concept of the hybrid methods.

1.4.1. Glimm's method. In surface tracking methods, the interface is represented with set of marker points and interpolation between them. Glimm et al developed a simulation package based on this approach (look at [**15**] and references within). This algorithm is a combination of a grid-base and a grid-free schemes. A marker particle method in which particles are located on the interface is used. They proposed to update the front dynamics with the grid-free method and the fluid dynamics away from the interface with a grid based method. This hybrid method, called the *locally grid-base method*, converts the grid-free representation of interface to the grid-base model to apply the grid-base method in bifurcation regions. Their comparison with other methods showed the superior results compare to volume-of-fluid and level set methods. However, no comparison made with the above mentioned finite volume method.

1.4.2. Peskin's method. Peskin developed the Immersed Boundary (IB) method to simulate cardiac mechanics and blood flow dynamics around heart (for references and applications see the review [**26**]). A method was developed in which a Cartesian grid is used for simulation with an Eulerian method. However, the interface is represented with a set of elastic fibers whose orientation is determined by massless particles which are evolved with a Lagrangian method, i.e., the particles velocity is the velocity of the surrounding fluid. The front-fluid interaction is completed by the force from a constitutive law such as Hooke's law. Since this force is pointwise and local, a smooth representation of Dirac delta function is used for coupling it with the fluid dynamics. Such methods are widely applied in the cases that interface geometry is complex such as simulation of biological systems, since the geometry representation is decoupled from the general grid structure.

1.5. Embedded boundary methods

Although embedded boundary methods do not consider the moving boundaries directly, the approaches they use to handle the complex geometries are inspiring sources to come

up with extensions to those do resolve moving boundary and free interface problems. The derivation of [**17**] from the embedded boundary method of [**14**] is an example of such an approach.

1.5.1. Embedded boundary method on a Cartesian grid. The method introduced in [**14**] applies the same kind of idea as [**3, 7**] to resolve the small-cell instability problem by using a combination of conservative and non-conservative but stable schemes on irregular control volumes and redistributing the difference in mass to neighboring cells. The interesting part of this method is the approach to find the fluxes on the covered edges and the algorithms for extrapolating and interpolating variables. The resulting method is second order accurate for smooth boundaries.

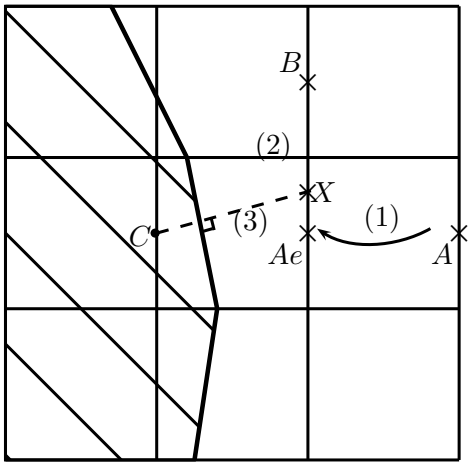

FIGURE 1.5. Extrapolation of variables on a covered face. (0) The point X is found as the first intersection of the normal to the boundary from C with another face in the same direction as covered face. (1) Variables at Ae are extrapolated from adjacent cell. (2) Variables at X are interpolated from variables at Ae and B. (3) Variables at C are calculated by extrapolation along the normal from X using the second order slopes.

1.5.2. Geometry by recursive application of the divergence theorem. This method concerns the computation of conservation form PDEs that requires a flux term on a boundary that does not conforms to the computational grid [**21**]. This introduces irregular computational cells that are cut by the embedded boundary. Writing the divergence theorem for the flux terms, one may expand the fluxes using the Taylor expansion around the appropriate centroids to have a definition of flux divergence based on integrals of polynomials over the volume and surface of the control volume. Putting this definition

in the divergence theorem along with the expansion of the surface normal vector results in a recursive method that in each recursion step reduces the geometrical dimension of the problem while keeping the order of accuracy intact. The solution in one dimension may be found explicitly and therefore solve the original high dimensional problem.

1.6. Method criteria

As described above, several methods have been developed for modeling a system with a moving interface. They have similarities and differences in every aspect. Here we want to choose a method for simulation of a moving interface in 1D while keeping in mind the goal of simulation in higher dimensions. For this purpose the following features are considered.

1.6.1. Domain representation. We covered methods which applied Cartesian grids and unstructured grids. For a non-Cartesian grid method that conforms to the interface, the re-griding procedure to overcome mesh entanglement is a problem which only adds complexity to the algorithm. Also for a fixed grid non-Cartesian Lagrangian method, e.g. [**8, 9**], the theoretical approach for analysis of error and accuracy is more complicated than for an Eulerian, Cartesian grid method. In addition, implementing a Cartesian grid method is less intricate. Also, a Cartesian grid is more suitable for applying the adaptive mesh refinement (AMR). Therefore we prefer to use a Cartesian grid method.

1.6.2. Interface representation. We like to use a general method to represent the moving boundary to be able to use features of other specialized methods. Therefore, we prefer to represent the front by its intersections with the Cartesian grid. One may extract the volume-of-fluid data form this representation or convert it to level set data and use the corresponding features. This representation also permits us to apply adaptive mesh refinement algorithms to resolve the fine dynamics in the system as applied in [**3, 23, 24**].

1.6.3. Integration method. We are aiming to achieve second order global accuracy at least for smooth initial conditions. The aforementioned methods generally cover this goal. Therefore, the main concern is the accuracy and stability at the interface. Here we prefer applying a method that gets its accuracy with a precise treatment of front geometry and states. This is done precisely in [**17**] with special treatment of centered and centroid states in the Cartesian cells.

In conclusion, the front tracking method of Gatti-Bono et al fulfill the above criteria to a great extent. As mentioned before, this method is based on [**14**] which will be a helpful source for extension of the method to higher dimensions.

CHAPTER 2

A front-tracking finite volume method in 1D

In this chapter the front-tracking method of Gatti-Bono et al [**17**] is explained in detail.

2.1. Gas dynamics and geometry description

Based on the definition in (1.1), a hyperbolic system of conservation laws in one dimension is described as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0. \tag{2.1}$$

Considering the Euler equations for compressible and inviscid fluids, conserved variables and corresponding fluxes are

$$\mathbf{U} = (\rho,\, \rho u,\, E)^T\,, \tag{2.2}$$

$$\mathbf{F} = \left(\rho u,\, \rho u^2 + p, (E+p)u\right)^T\,, \tag{2.3}$$

where ρ, u and p are density, velocity and pressure of fluid respectively. The equation of state for an ideal polytropic gas completes this system of equations by stating the relation between total energy E and conserved variables,

$$E \equiv \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2, \tag{2.4}$$

where γ is the adiabatic exponent of the fluid.

Considering only one front in this problem, fluids on both sides of the front are governed by the conservation equation stated in (2.1)-(2.4). The front is identified by its position x^f and its speed $s^f = \frac{dx^f}{dt}$. Also the fluids on both sides of the front fulfill the Rankine-Hugoniot condition [**20**],

$$\left[\left(\mathbf{F} - s^f\mathbf{U}\right).\mathbf{n}\right]_L - \left[\left(\mathbf{F} - s^f\mathbf{U}\right).\mathbf{n}\right]_R = 0, \tag{2.5}$$

where states of left and right fluids are shown with L and R respectively.

The definition of the system in primitive variables is required for the solution of the Riemann problem,

$$\frac{\partial \mathbf{W}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{W}}{\partial x} = 0, \tag{2.6}$$

where

$$\mathbf{W} = (\rho,\, u,\, p)^T, \tag{2.7}$$

$$\mathbf{A} = \begin{pmatrix} u & \rho & 0 \\ 0 & u & \frac{1}{\rho} \\ 0 & \gamma p & u \end{pmatrix}. \tag{2.8}$$

The time-space domain is discretized with a regular Cartesian grid with spacing Δx and Δt in space and time respectively. The description of space can be generalized by defining Υ_i as the ith Cartesian grid cell and Ω as the problem domain. Therefore, each control volume may be represented by $V_i = \Upsilon_i \cap \Omega$ and its volume as $|V_i| = \int_{V_i} \mathrm{d}x$. The volume fraction κ_i is the ratio of ith cell volume and that of a regular cell. Similarly, the area fraction for a face α is defined as the ratio of length of that face and that of a regular face. To provide appropriate data for a finite volume method, discretized conserved variables are defined at space centroids and at discretized time steps, $\overline{\mathbf{U}}_i^n = \mathbf{U}\left(x_{i,c}^n, t^n\right)$. Also, fluxes are calculated on the grid faces and time centroids, shown as $\overline{\mathbf{F}}$. Primitive variables are evaluated on the spacial center of faces and at half time steps. A schematic view of grid and front is shown in Figure 2.1.

2.2. Discretization method

To derive the finite volume method, the divergence theorem is applied to (2.1) in time and space,

$$\int_{V_i(t)} \left(\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x}\right) \mathrm{d}x\mathrm{d}t = 0, \tag{2.9}$$

where

$$V_i(t) = [t^n; t^{n+1}] \times (\Omega^a \cap \Upsilon_i), \qquad a \in \{L, R\}. \tag{2.10}$$

The Reynolds transport theorem [**11**], the generalization of Leibniz integral rule [**16**], states the formula for differentiation under integral sign for a variable volume,

$$\frac{\partial}{\partial t}\int_{\Omega(t)} U(\mathbf{x},t)\,\mathrm{d}\mathbf{x} = \int_{\Omega(t)} \frac{\partial U}{\partial t}\,\mathrm{d}\mathbf{x} + \oint_{\partial\Omega(t)} U s^f\,\mathrm{d}A. \tag{2.11}$$

Separating the integral in (2.9), applying (2.11) to exchange the space integral and time differentiation, and substituting the volume integral with a surface integral using the divergence theorem result in

$$\int_{t^n}^{t^{n+1}} \left(\frac{\partial}{\partial t}\int_{\Omega^a(t)\cap\Upsilon_i} \mathbf{U}\mathrm{d}x - \oint_{\partial[\Omega^a(t)\cap\Upsilon_i]} \mathbf{U} s^f\,\mathrm{d}A. \right) \mathrm{d}t + \oint_{\partial V_i(t)} \mathbf{F}.\mathbf{n}\,\mathrm{d}S = 0, \tag{2.12}$$

where $\mathbf{n}$ is the outward normal vector at the surface. Decomposing the integration domain of the last integral in (2.12) into two parts, the front $\mathcal{F}(t)$ and the rest $\partial V_i(t)\backslash\mathcal{F}(t)$, gives

$$\left[\int_{\Omega^a(t)\cap\Upsilon_i} \mathbf{U}\mathrm{d}x\right]_{t^n}^{t^{n+1}} + \int_{\partial V_i(t)\backslash\mathcal{F}(t)} \mathbf{F}.\mathbf{n}\,\mathrm{d}S + \int_{\mathcal{F}(t)} \left(\mathbf{F} - s^f\mathbf{U}\right).\mathbf{n}\,\mathrm{d}S = 0. \tag{2.13}$$

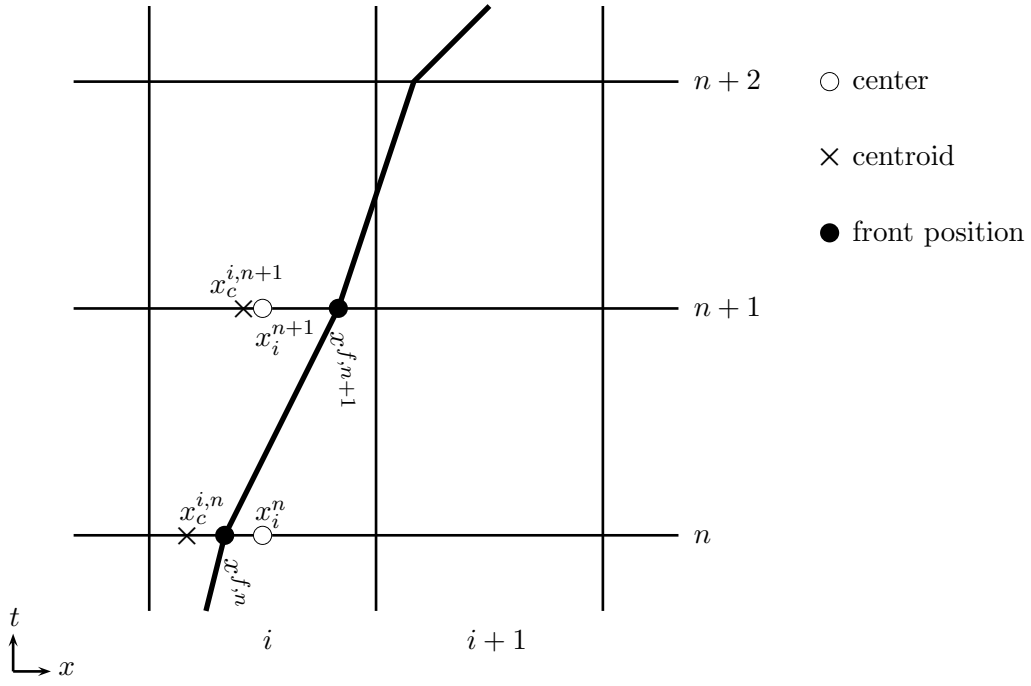

FIGURE 2.1. Discretization in space and time with definition of the front on the Cartesian grid

Discretizing the equation (2.13) in time and space results in

$$\Delta x \left(\kappa_{a,i}^{n+1}\overline{\mathbf{U}}_{a,i}^{n+1} - \kappa_{a,i}^{n}\overline{\mathbf{U}}_{a,i}^{n}\right) + \Delta t \left(\alpha_{a,i+\frac{1}{2}}\overline{\mathbf{F}}_{a,i+\frac{1}{2}} - \alpha_{a,i-\frac{1}{2}}\overline{\mathbf{F}}_{a,i-\frac{1}{2}} \mp \alpha^{f}\overline{\mathbf{F}}^{f}\right) = 0, \tag{2.14}$$

where

$$\overline{\mathbf{F}}^{f} = \overline{\mathbf{F}} - s^{f}\overline{\mathbf{U}}, \tag{2.15}$$

$$\overline{\mathbf{U}}_{a,i}^{n} \approx \frac{1}{\kappa_{a,i}^{n}\Delta x}\int_{[x_{i-\frac{1}{2}};x_{i+\frac{1}{2}}]} \mathbf{U}(x, n\Delta t)\,\mathrm{d}x, \qquad \forall \kappa_{a,i}^{n}\, 0. \tag{2.16}$$

One can define $\overline{\mathbf{U}}^{n,n+1}$ as the conserved variable evaluated at time t^n at the centroid position of time t^{n+1},

$$\overline{\mathbf{U}}^{n,n+1} = \mathbf{U}(x_c^{n+1}, t^n). \tag{2.17}$$

To obtain a convergent discretization, one can modify (2.14) to develop the update equation as follows:

$$\kappa_{a,i}^{n+1}\left(\overline{\mathbf{U}}_{a,i}^{n+1} - \overline{\mathbf{U}}_{a,i}^{n,n+1}\right) = \kappa_{a,i}^{n}\overline{\mathbf{U}}_{a,i}^{n} - \kappa_{a,i}^{n+1}\overline{\mathbf{U}}_{a,i}^{n,n+1} - \frac{\Delta t}{\Delta x}\left(\alpha_{a,i+\frac{1}{2}}\overline{\mathbf{F}}_{a,i+\frac{1}{2}} - \alpha_{a,i-\frac{1}{2}}\overline{\mathbf{F}}_{a,i-\frac{1}{2}} \pm \alpha^{f}\overline{\mathbf{F}}^{f}\right). \tag{2.18}$$

Defining the conservative flux difference based on the right hand side of (2.18)

$$\overline{DF}_{a,i}^{C} = \frac{1}{\kappa_{a,i}^{n+1}}\left(\frac{\kappa_{a,i}^{n+1}\overline{\mathbf{U}}_{a,i}^{n,n+1} - \kappa_{a,i}^{n}\overline{\mathbf{U}}_{a,i}^{n}}{\Delta t} + \frac{\alpha_{a,i+\frac{1}{2}}\overline{\mathbf{F}}_{a,i+\frac{1}{2}} - \alpha_{a,i-\frac{1}{2}}\overline{\mathbf{F}}_{a,i-\frac{1}{2}} \pm \alpha^{f}\overline{\mathbf{F}}^{f}}{\Delta x}\right), \tag{2.19}$$

one may write the update equation as

$$\overline{\mathbf{U}}_{a,i}^{n+1} = \overline{\mathbf{U}}_{a,i}^{n,n+1} - \Delta t\overline{DF}_{a,i}^{C}, \tag{2.20}$$

which is more convenient physically, since the $\frac{\partial \mathbf{U}}{\partial t}$ is evaluated on the same position in space. Similarly, the non-conservative flux difference is defined as follows:

$$DF_{a,i}^{NC} = \frac{\overline{\mathbf{F}}_{a,i+\frac{1}{2}} - \overline{\mathbf{F}}_{a,i-\frac{1}{2}}}{\Delta x}. \tag{2.21}$$

As discussed in the previous chapter, the conservative flux difference becomes unstable for irregular cells with small volume fractions. Therefore, using approaches similar to [**3, 7, 23**], the update equation exploits a weighted average of conservative and non-conservative flux

differences,

$$\overline{\mathbf{U}}_{a,i}^{n+1} = \overline{\mathbf{U}}_{a,i}^{n,n+1} - \Delta t \left[\kappa_{a,i}^{n+1} \overline{DF}_{a,i}^{C} + \left(1 - \kappa_{a,i}^{n+1}\right) \overline{DF}_{a,i}^{NC} \right]. \tag{2.22}$$

To satisfy the global conservation, the mass difference between conservative and non-conservative updates is redistributed to the neighbor cells. The mass difference is defined as follows:

$$\delta M_{a,i} = \kappa_{a,i}^{n+1} \left[\overline{DF}_{a,i}^{C} - \left(\kappa_{a,i}^{n+1} \overline{DF}_{a,i}^{C} + \left(1 - \kappa_{a,i}^{n+1}\right) \overline{DF}_{a,i}^{NC} \right) \right] \Delta t. \tag{2.23}$$

2.3. Algorithm - 1D

After initialization at the beginning of each update loop, $\overline{\mathbf{U}}_a^n$ and $x^{f,n}$ are given and the goal is to find $\overline{\mathbf{U}}_a^{n+1}$ and $x^{f,n+1}$ (Figure 2.2). The overview of the algorithm for the computation loop based on the update equation (2.22) is as follows:

- Interface update
  - Evaluate front speed at time t^n
  - Advance to time t^{n+1} using non-conservative fluxes (2.21)
  - Evaluate front speed at time t^{n+1}
  - Update interface using front speed at $t^{n+\frac{1}{2}}$
- Conserved variable update
  - Non-conservative flux difference (2.22)
  - Conservative flux difference (2.19)
- Mass difference redistribution (2.23)

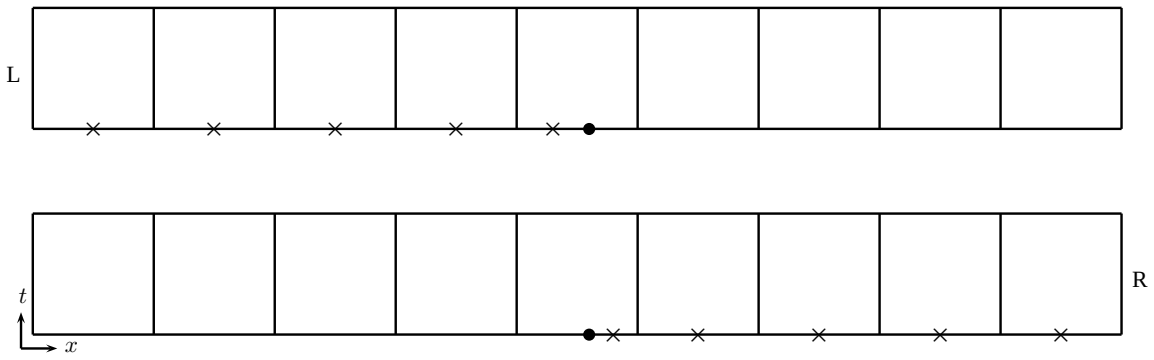

FIGURE 2.2. Initial data available at the beginning of each update loop for left and right fluids and interface.

2.3.1. Spatially centered and centroid variables. The second order flux calculation in this algorithm requires the second order accurate state variables at space centers $U^n_{a,i}$. For regular cells, $\overline{U}^n_{a,i}$ approximates $U^n_{a,i}$ to $O(h^2)$. However, for an irregular cell, that is $i = i^f$, extrapolation is needed to find U^n_{a,i^f}. One may use the extrapolation of states at space centroids on the front cell and its adjacent regular cell to find the second order approximation of U^n_{a,i^f}. For an irregular cell and the side belonging to the left fluid, it is derived as follows:

$$U^n_{L,i^f} = \overline{U}^n_{L,i^f} + \frac{\overline{U}^n_{L,i^f} - \overline{U}^n_{L,i^f-1}}{\frac{1}{2} + \frac{\kappa^n_{L,i^f}}{2}} \left(\frac{1}{2} - \frac{\kappa^n_{L,i^f}}{2} \right). \tag{2.24}$$

Therefore, for both left and right fluids, one may write,

$$U^n_{a,i^f} = \overline{U}^n_{a,i^f} + \frac{1 - \kappa^n_{a,i^f}}{1 + \kappa^n_{a,i^f}} \left(\overline{U}^n_{a,i^f} - \overline{U}^n_{a,i^f \mp 1} \right). \tag{2.25}$$

This procedure is depicted in Figure 2.3.

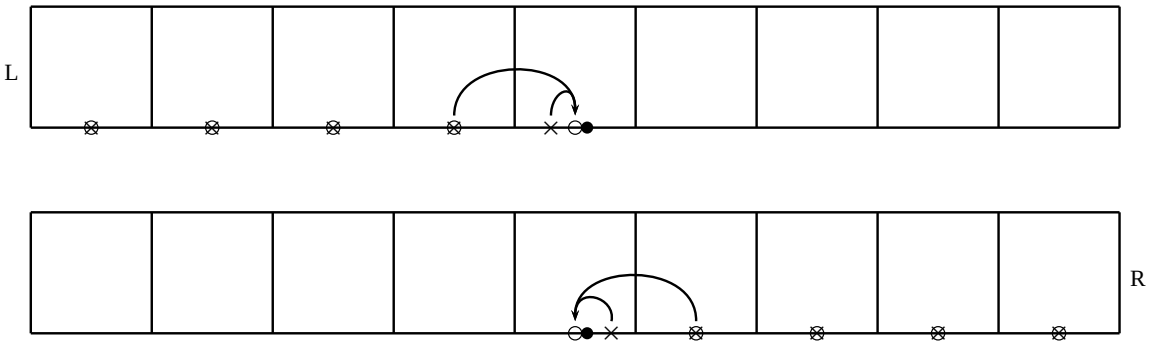

FIGURE 2.3. Second order extrapolation from centroids to centers for cells at front. For regular cells centers and centroids are at the same positions.

2.3.2. Interface update. To advance the front position from $x^{f,n}$ to $x^{f,n+1}$ the velocity of front is needed. The second order approximation of the front speed is estimated by the average value of the front speed at times t^n and t^{n+1}. The front speed at time t^n can be evaluated from solution of Riemann problem on the front. The left and right states on the front at t^n are computed by second order extrapolation of the cell centered values on the front cell and its adjacent cell. For the left initial state it is derived as follows:

$$W^n_{L,x^f} = W^n_{L,i^f} + \left(W^n_{L,i^f} - W^n_{L,i^f-1} \right) \left(\kappa^n_{L,i^f} - \frac{1}{2} \right), \tag{2.26}$$

where $W^n_{a,i} = W(U^n_{a,i})$. Consequently, for both left and right fluids it states

$$W^n_{a,x^f} = \frac{1}{2}\left(W^n_{a,i^f} + W^n_{a,i^f\mp 1}\right) + \kappa^n_{a,i^f}\left(W^n_{a,i^f} - W^n_{a,i^f\mp 1}\right), \tag{2.27}$$

shown in Figure 2.4.

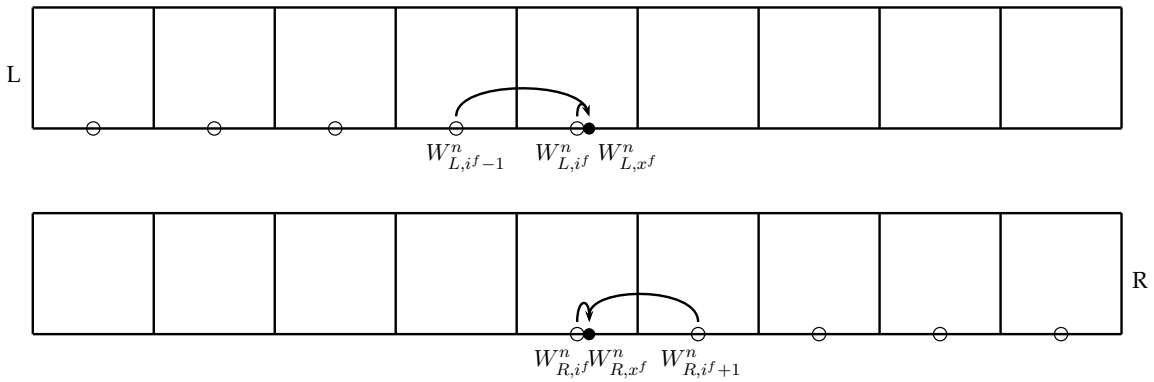

FIGURE 2.4. Extrapolation to the front for the initial value of Riemann problem to find the front speed.

Therefore, the estimation of front speed at time t^n is

$$s^{f,n} = s\left(\mathcal{R}\left(W^n_{L,x^f}, W^n_{R,x^f}\right)\right). \tag{2.28}$$

Then, the interface is advanced to t^{n+1} using the first estimate of the front speed.

$$x^{f,n+1} = x^{f,n} + \Delta t s^{f,n}. \tag{2.29}$$

The front may move from one cell to another during the temporal update. Therefore it is necessary to estimate the state variables at the new uncovered cell. This can be done by adding the limited spacial differences to the conserved variables (Figure 2.4).

$$\mathbf{U}^n_{i^{f^{n+1}}} = \mathbf{U}^n_{i^{f^n}} \pm \Delta\mathbf{U}, \tag{2.30}$$

where

$$\Delta\mathbf{U} = \begin{cases} min\left(2\left|\Delta U_{i^{f^n}\mp\frac{1}{2}}\right|, 2\left|\Delta U_{i^{f^n}\mp\frac{3}{2}}\right|, 2\left|\Delta U_{i^{f^n}\mp\frac{5}{2}}\right|, \left|\Delta U_{i^{f^n}\mp 2}\right|\right) \\ \quad \text{if } \Delta U_{i^{f^n}\mp\frac{1}{2}}\Delta U_{i^{f^n}\mp\frac{3}{2}} > 0, \Delta U_{i^{f^n}\mp\frac{3}{2}}\Delta U_{i^{f^n}\mp\frac{5}{2}} > 0, \\ 0 \quad \text{otherwise,} \end{cases}$$

with

$$\Delta U_{i\mp\frac{1}{2}} = U_i - U_{i\mp 1}, \quad \Delta U_{i\mp 2} = \frac{U_{i\mp 1} - U_{i\mp 3}}{2}. \tag{2.31}$$

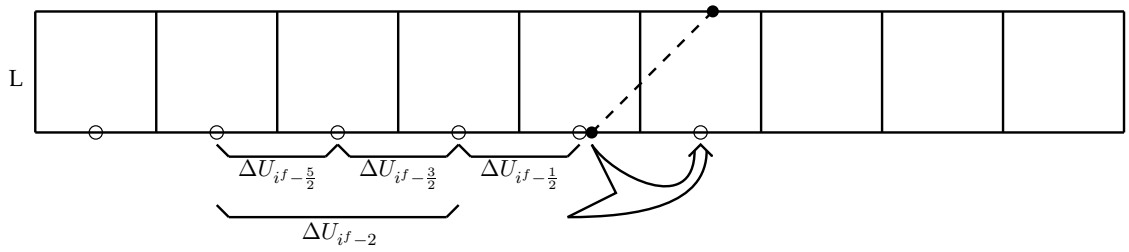

FIGURE 2.5. Primary calculation of front at time $n+1$ based of $s^{f,n}$ and value estimation for newly uncovered cell, if front changes cell after one time step.

To derive the front speed at time t^{n+1}, the solution is advanced using the non-conservative fluxes:

$$\tilde{\mathbf{U}}_{a,i}^{n+1} = \mathbf{U}_{a,i}^{n} - \Delta t \, \overline{DF}_{a,i}^{NC}, \tag{2.32}$$

where $\overline{DF}_{a,i}^{NC}$ is defined in (2.21). Fluxes for this computation are evaluated by the primitive variables that are advanced to half time step (this procedure is described in detail in the next subsection),

$$\mathbf{F}_{a,i+\frac{1}{2}} = \mathbf{F}\left(\tilde{\mathbf{W}}_{a,i+\frac{1}{2}}\right). \tag{2.33}$$

The left and right states for the Riemann problem at time t^{n+1} are evaluated based on the approximate solution, $\tilde{W}_{a,i}^{n+1} = W\left(\tilde{U}_{a,i}^{n+1}\right)$, and the extrapolation procedure in (2.27). The speed is defined form the solution of the Riemann problem,

$$s^{f,n+1} = s\left(\mathcal{R}\left(\tilde{W}_{L,x^{f,n+1}}^{n+1}, \tilde{W}_{R,x^{f,n+1}}^{n+1}\right)\right). \tag{2.34}$$

Finally, the front position is updated using the second order estimation of front speed,

$$s^{f} = \frac{s^{f,n} + s^{f,n+1}}{2}, \tag{2.35}$$

$$x^{f,n+1} = x^{f,n} + \Delta t s^{f}. \tag{2.36}$$

2.3.3. Non-conservative flux difference. The update equation (2.22) has two main parts, the conservative and non-conservative flux difference. For the non-conservative but stable part, fluxes at time centers are needed. These fluxes are evaluated by finding the primitive variables at those points as the solution of a Riemann problem with a standard Godunov method [**12**]. From the center of each cell, a vector of primitive variables $\mathbf{W}$ is extended to the faces to create $\mathbf{W}^{+}$ and $\mathbf{W}^{-}$; the left and right states on those faces. For

a regular cell this procedure reads

$$\mathbf{W}^{\pm}_{a,i}\left(x_i \pm \frac{\Delta x}{2}; t^n + \frac{\delta t}{2}\right) = \mathbf{W}^{n}_{a,i} + \frac{1}{2}\sum_k \eta^k \left(\pm - \frac{\delta t}{\Delta x} max(\pm\lambda^k, 0)\right)\mathbf{r}^k, \tag{2.37}$$

where

$$\delta t = \begin{cases} \Delta t & \text{to derive } W \text{ at half time step,} \\ 0 & \text{to derive } W \text{ current time step,} \end{cases} \tag{2.38}$$

and λ^k,$\mathbf{r}^k$ and $\mathbf{l}^k$ are the kth eigenvalue, right and left eigenvectors of matrix A respectively. For a regular face, both positive and negative states are accessible from left and right cells. However, for a face at the boundary or a covered face, one of the states may not be available since there is no cell on that side. This state is extrapolated from the same sign of states of the two nearest adjacent cell with second order extrapolation (Figure 2.6).

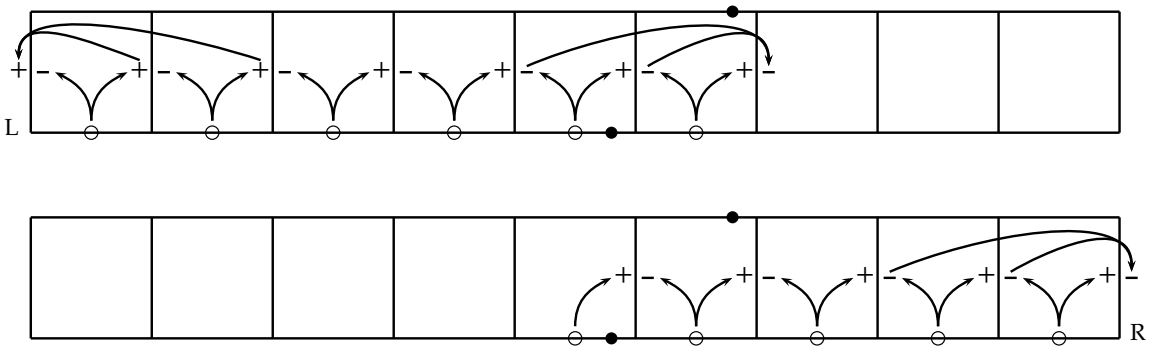

FIGURE 2.6. Computation of $W^{\pm}$ for regular and irregular (at the boundary or covered) faces.

In (2.37), η^k is the coefficient for the expansion of the slope over the space spanned by left eigenvectors. This slope is also limited using the van Leer limiter to avoid introducing unphysical oscillations:.

$$\Delta^v \mathbf{W} = \sum_k \eta^k \mathbf{r}^k, \tag{2.39}$$

where

$$\eta^k = \begin{cases} \min(|\eta^k_L|, |\eta^k_R|, |\eta^k_C|)\,\mathrm{sgn}(\eta^k_C) & \text{if } \eta^k_L \eta^k_R > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{2.40}$$

and

$$\eta_L^k = 2\mathbf{l}^k.\Delta\mathbf{W}_L, \quad \Delta\mathbf{W}_L = \mathbf{W}_i - \mathbf{W}_{i-1}, \tag{2.41}$$

$$\eta_R^k = 2\mathbf{l}^k.\Delta\mathbf{W}_R, \quad \Delta\mathbf{W}_R = \mathbf{W}_{i+1} - \mathbf{W}_i, \tag{2.42}$$

$$\eta_C^k = \frac{1}{2}\mathbf{l}^k.\Delta\mathbf{W}_C, \quad \Delta\mathbf{W}_C = \mathbf{W}_{i+1} - \mathbf{W}_{i-1}. \tag{2.43}$$

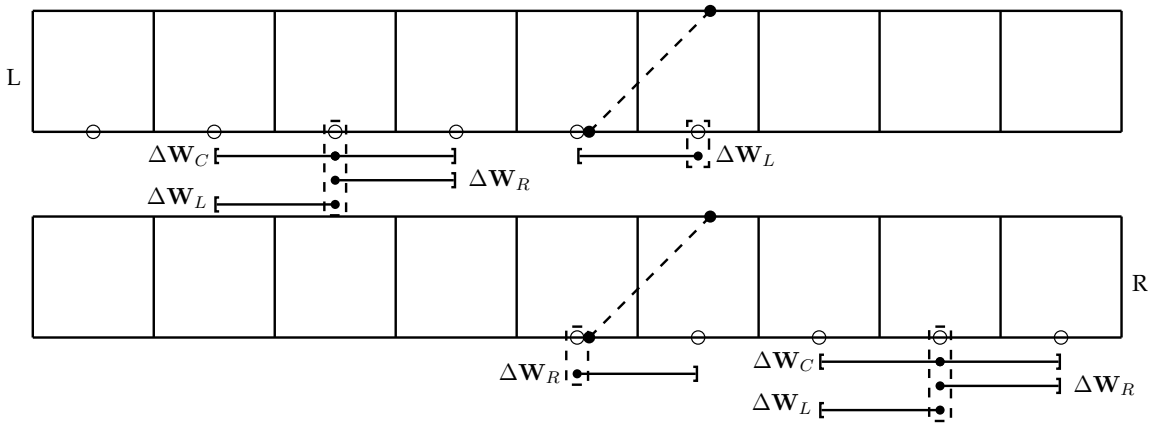

FIGURE 2.7. Slope calculation for applying Van Leer limiter. For a cell in the middle of the fluid domain all one-sided and centered differences are available but for cell near to the boundary or front some of those differences may not be available.

For a covered face, the above procedure calculates the state which comes from only one of the fluids. In other words, that state is not affected by jumping over the front, which is unphysical. To overcome this problem, the left and right states on the covered face are extrapolated back to the front from the state variables of each fluids. That is, the left fluid provides the left initial state of the Riemann problems and the right fluid provides the right corresponding state. After solving the Riemann problem, the result is extrapolated back to the covered face. This procedure for the state of the covered face from left fluid point of view is shown in Figure 2.8 and reads as follows:

$$W_{LL}(x^f, t^{n+\frac{1}{2}}) = W_{L,i+\frac{1}{2}} + \frac{x^f - x_{i+\frac{1}{2}}}{\Delta x}\Delta W_L, \tag{2.44}$$

$$W_{RL}(x^f, t^{n+\frac{1}{2}}) = W_{R,i+\frac{1}{2}} + \frac{x^f - x_{i+\frac{1}{2}}}{\Delta x}\Delta W_R. \tag{2.45}$$

The solution of the Riemann problem at the front gives

$$W_L(x^f, t^{n+\frac{1}{2}}) = \mathcal{R}\left(W_{LL}, W_{RL}\right), \tag{2.46}$$

which is extrapolated back to the face,

$$W_{L,i+\frac{1}{2}} = W_L(x^f, t^{n+\frac{1}{2}}) - \frac{x^f - x_{i+\frac{1}{2}}}{\Delta x}\Delta W_L. \tag{2.47}$$

With all states available in half time step, fluxes are computed and the non-conservative

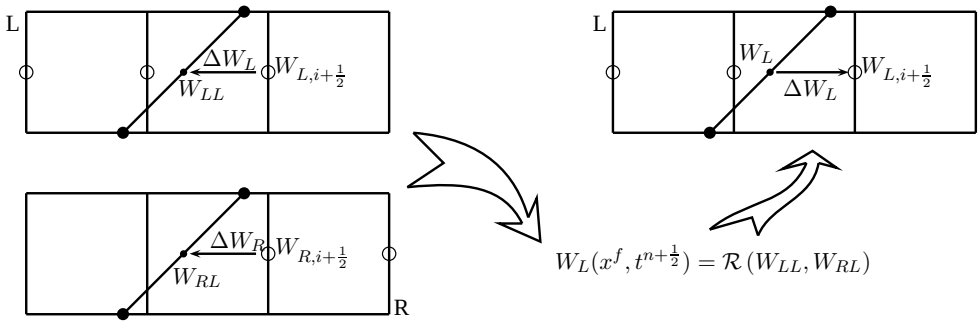

FIGURE 2.8. Extrapolation on the covered face while applying jump condition for left fluid.

flux difference is evaluated from (2.21). For the front cell, this value is cell centered and is averaged to cell centroid before evaluating the update equation (2.22):

$$\overline{DF}^{NC}_{a,i} = \left(DF^{NC}_{a,i}\right)^{CC\to Ce}. \tag{2.48}$$

2.3.4. Conservative flux difference. To compute the conservative flux difference from (2.19), fluxes are needed at the centroid of the time-space aperture and the front (Figure 2.9a).

As shown in Figure 2.9b, on a face crossed by the front, the vector of primitive variables at time centroid $\mathbf{W}^{n_c}_{a,i+\frac{1}{2}}$ is extrapolated using primitive variables at the half time step $\mathbf{W}^{n+\frac{1}{2}}_{a,i+\frac{1}{2}}$ and the current time step $\mathbf{W}^{n}_{a,i+\frac{1}{2}}$,

$$\mathbf{W}^{n_c}_{a,i+\frac{1}{2}} = \mathbf{W}^{n+\frac{1}{2}}_{a,i+\frac{1}{2}} \pm (1-\alpha_a)\left(\mathbf{W}^{n+\frac{1}{2}}_{a,i+\frac{1}{2}} - \mathbf{W}^{n}_{a,i+\frac{1}{2}}\right). \tag{2.49}$$

On the front, states from centered variables are extrapolated to the centroids on the front as the left and right initial values of the Riemann problem. Solution of the Riemann problem gives the centroid primitive variables on the front (Figure 2.9c). Then the flux on the front

is computed exactly,

$$\overline{\mathbf{F}}^{f} = (0,\, p,\, up)^{T}\,. \tag{2.50}$$

Then the conservative flux difference is calculated by

$$\overline{DF}^{C}_{a,i} = \frac{1}{\kappa^{n+1}_{a,i}}\left(\frac{\kappa^{n+1}_{a,i}\overline{\mathbf{U}}^{n,n+1}_{a,i} - \kappa^{n}_{a,i}\overline{\mathbf{U}}^{n}_{a,i}}{\Delta t} + \frac{\alpha_{a,i+\frac{1}{2}}\overline{\mathbf{F}}_{a,i+\frac{1}{2}} - \alpha_{a,i-\frac{1}{2}}\overline{\mathbf{F}}_{a,i-\frac{1}{2}} \pm \alpha^{f}\overline{\mathbf{F}}^{f}}{\Delta x}\right). \tag{2.51}$$

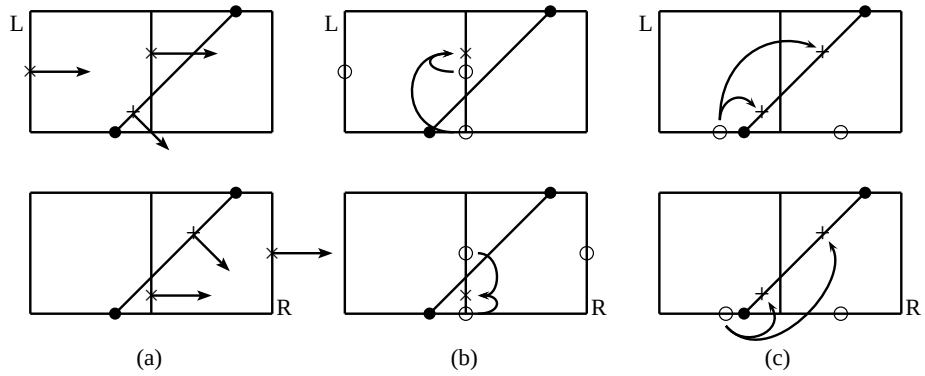

FIGURE 2.9. (a)Fluxes needed for conserved flux difference of front cell. (b) Extrapolation of time centroid states on a face crossed by the front. (c) Extrapolation to the front to calculate the initial states to solve the Riemann problem on front centroids.

2.3.5. Mass difference redistribution. The mass difference between the hybrid update method (2.22) and conservative update (2.20) should be redistributed to ensure the conservation of the method. One way to achieve this is to project the excessive mass on the eigenvectors to redistribute based on characteristics:

$$\delta M_a = -\Delta t\,\kappa^{n+1}_{a,i}\left(1-\kappa^{n+1}_{a,i}\right)\left[\overline{DF}^{C}_{a,i} - \overline{DF}^{C}_{a,i}\right] = \sum_{k}\beta^{k}_{a}\mathbf{r}^{k}_{a}, \tag{2.52}$$

where $k \in \{-1, 0, 1\}$, indicate the characteristics from left to right based on their speed. Therefore for the front which resemble a contact discontinuity with acoustic impedance

$$\mu_L = \frac{1/\rho_L c_L}{1/\rho_L c_L + 1/\rho_R c_R}, \tag{2.53}$$

$$\mu_R = \frac{1/\rho_R c_R}{1/\rho_L c_L + 1/\rho_R c_R}, \tag{2.54}$$

the redistributed mass is defined as follows:

$$\delta\mathbf{M}_L^{red} = \beta_L^0\mathbf{r}_L^0 + \beta_L^{-1}\mathbf{r}_L^{-1} + \mu_R\beta_R^{-1}\mathbf{r}_R^{-1} + (1-\mu_L)\beta_L^1\mathbf{r}_L^1, \tag{2.55}$$

$$\delta\mathbf{M}_R^{red} = \beta_R^0\mathbf{r}_R^0 + \beta_R^1\mathbf{r}_R^1 + (1-\mu_R)\beta_R^{-1}\mathbf{r}_R^{-1} + \mu_L\beta_L^1\mathbf{r}_L^1. \tag{2.56}$$

The states are then reupdated for the front cell and the adjacent cells,

$$\overline{\mathbf{U}}_{a,i}^{n+1} = \overline{\mathbf{U}}_{a,i}^{n+1} + \frac{1}{1+\kappa_a^{n+1}}\delta\mathbf{M}_a^{red}, \tag{2.57}$$

where

$$i \in \begin{cases} \{i^f, i^f - 1\} & a = L, \\ \{i^f, i^f + 1\} & a = R, \end{cases} \tag{2.58}$$

and $i^f = i(x^f, t^{n+1})$.

CHAPTER 3

Proposal for the front-tracking method in 2D

In this chapter, a proposal is given for the extension to 2D of the method described in the previous chapter. Adding one dimension in space brings issues in interface update, extrapolation and integration method which are different from the 1D case and need extra attention. A brief description of the problem in 2D, an overview to the algorithm and details about some parts of the algorithm are presented here.

3.1. Gas dynamics and geometry description

Here, we use the gas dynamics in 2D as defined in (1.1) with $D = 2$ and no source term. Considering the Euler equations for compressible and inviscid fluids, conserved variables and corresponding fluxes are

$$\mathbf{U} = (\rho,\, \rho u,\, \rho v,\, E)^T\,, \tag{3.1}$$

$$\mathbf{F}^1 = \left(\rho u,\, \rho u^2 + p,\, \rho uv,\, (E+p)u\right)^T\,, \tag{3.2}$$

$$\mathbf{F}^2 = \left(\rho v,\, \rho uv,\, \rho v^2 + p,\, (E+p)v\right)^T\,, \tag{3.3}$$

where u, v are velocity in the x and y direction respectively, and total energy E is defined as

$$E \equiv \frac{p}{\gamma - 1} + \frac{1}{2}\rho\left(u^2 + v^2\right). \tag{3.4}$$

We only consider one interface for the simplicity of the algorithm. Both phases on each side of the interface are governed by the conservation equations stated above. The interface is a curve in 2D given by $\mathcal{F}(t)$. For each point on the interface $\mathbf{x}^f$, speed of the front is $\mathbf{s}^f = \frac{d\mathbf{x}^f}{dt}$. The interface speed in normal direction is defined as $v^f(\mathbf{x}^f(t)) = \mathbf{s}^f.\mathbf{n}$, where $\mathbf{n}$ is the normal unit vector on the interface from phase 1 to 2. Across the interface the Rankine-Hugoniot conditions apply in the normal direction, that is, equation (2.5) applies to the dynamics rotated in the normal direction to the interface.

The definition of the system in primitive variables is

$$\frac{\partial \mathbf{W}}{\partial t} + \sum_{d=1}^{D} \mathbf{A}^d \frac{\partial \mathbf{W}}{\partial x_d} = 0, \tag{3.5}$$

where for $D = 2$,

$$\mathbf{W} = (\rho,\, u,\, v,\, p)^T, \tag{3.6}$$

$$\mathbf{A}^1 = \begin{pmatrix} u & \rho & 0 & 0 \\ 0 & u & 0 & \frac{1}{\rho} \\ 0 & 0 & u & 0 \\ 0 & \gamma p & 0 & u \end{pmatrix}, \qquad \mathbf{A}^2 = \begin{pmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & \frac{1}{\rho} \\ 0 & 0 & \gamma p & u \end{pmatrix}. \tag{3.7}$$

Similar to the 1D case, the time-space domain is discretized with a regular Cartesian grid with discretization h and Δt in space and time respectively. A cell of the Cartesian grid in space is defined as $\Upsilon_{\mathbf{i}} = [(\mathbf{i} - \frac{1}{2}\mathbf{v})h, (\mathbf{i} + \frac{1}{2}\mathbf{v})h]$, $\mathbf{i} \in \mathbb{Z}^D$ where $\mathbf{v}$ is the one vector. The control volume V_i is the intersection of the irregular domain Ω with $\Upsilon_{\mathbf{i}}$. Therefore faces of the control volume are labeled by $A_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$ as faces that coincide with the Cartesian grid, and $A^f_{\mathbf{i}}$ as part of control volume face that is on the front. Based on these definitions we generate geometrical properties for each cell. Volume fraction and area fractions are defined as follows:

$$\kappa_{\mathbf{i}} = \frac{|V_{\mathbf{i}}|}{h^D}, \qquad \alpha_{\mathbf{i}+\frac{1}{2}\mathbf{e}_s} = \frac{|A_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d_s}|}{h^{D-1}}, \qquad \alpha^f_{\mathbf{i}} = \frac{|A_{\mathbf{i}}|}{h^{D-1}}. \tag{3.8}$$

We define a level set function which we use for updating front procedure. Level set function $\psi(x,t)$ is defined as signed distance function respect to phase 1,

$$\psi(\mathbf{x}, t) = \pm l, \tag{3.9}$$

where l is the distance of point $\mathbf{x}$ to the front $\mathcal{F}(t)$ and plus (minus) is chosen if the point $\mathbf{x}$ is outside (inside) the phase 1 domain. This means that the zero level set depict the front position, $\psi(\mathcal{F}(t), t) = 0$.

3.2. Discretization method

Applying the same procedure as shown in (2.9)-(2.13) to the conservation laws in 2D results in

$$\left[\int_{\Omega^a(t)\cap\Upsilon_i}\mathbf{U}\mathrm{d}x\right]_{t^n}^{t^{n+1}}+\int_{\partial V_i(t)\setminus\mathcal{F}(t)}\mathbf{F}.\mathbf{n}\,\mathrm{d}S+\int_{\mathcal{F}(t)}\left(\mathbf{F}.\mathbf{n}-v^f\mathbf{U}\right)\,\mathrm{d}S=0. \tag{3.10}$$

Discretizing in time and space gives

$$h\left(\kappa_{a,\mathbf{i}}^{n+1}\overline{\mathbf{U}}_{a,\mathbf{i}}^{n+1}-\kappa_{a,\mathbf{i}}^{n}\overline{\mathbf{U}}_{a,\mathbf{i}}^{n}\right)+\Delta t\left(\sum_{\pm=+,-}\sum_{d=1}^{D}\pm\bar{\alpha}_{a,\mathbf{i}+\frac{1}{2}\mathbf{e}^d}\bar{\overline{F}}_{a,\mathbf{i}+\frac{1}{2}\mathbf{e}^d}+\bar{\alpha}_{\mathbf{i}}^{f}\bar{\overline{\mathbf{F}}}^{f}\right)=0, \tag{3.11}$$

where $a\in\{1,2\}$ and $\overline{q}$ (with thick, wide overline) and $\bar{q}$ (with thin, narrow bar) represent spatial volume-weighted and temporal aperture-weighted averages respectively,

$$\overline{\mathbf{U}}_{a,\mathbf{i}}^{n}\approx\frac{1}{\kappa_{a,\mathbf{i}}^{n}h^D}\int_{V_\mathbf{i}}\mathbf{U}(\mathbf{x},n\Delta t)\,\mathrm{d}x^D,\qquad\forall\kappa_{a,i}^{n}\,0, \tag{3.12}$$

$$\bar{\overline{\mathbf{F}}}^{f}=\bar{\overline{\mathbf{F}}}.\bar{\mathbf{n}}-\bar{v}^f\bar{\mathbf{U}}. \tag{3.13}$$

This means that fluxes are evaluated on space-time centroids of the aperture. Following the algorithm in 1D, we define the conservative and non-conservative flux differences,

$$\overline{DF}_{a,\mathbf{i}}^{C}=\frac{\kappa_{\mathbf{i}}^{n+1}\mathbf{U}_{\mathbf{i}}^{n,n+1}-\kappa_{\mathbf{i}}^{n}\mathbf{U}_{\mathbf{i}}^{n}}{\kappa_{\mathbf{i}}^{n+1}\Delta t}-\frac{1}{\kappa_{\mathbf{i}}^{n+1}h}\left(\sum_{\pm=+,-}\sum_{d=1}^{D}\pm\bar{\alpha}_{a,\mathbf{i}+\frac{1}{2}\mathbf{e}^d}\bar{\overline{F}}_{a,\mathbf{i}+\frac{1}{2}\mathbf{e}^d}+\bar{\alpha}_{\mathbf{i}}^{f}\bar{\overline{\mathbf{F}}}^{f}\right), \tag{3.14}$$

$$\overline{DF}_{a,\mathbf{i}}^{NC}=\frac{1}{h}\sum_{\pm=+,-}\sum_{d=1}^{D}\pm\bar{\overline{F}}_{a,\mathbf{i}+\frac{1}{2}\mathbf{e}^d}, \tag{3.15}$$

with the update equation

$$\overline{\mathbf{U}}_{a,\mathbf{i}}^{n+1}=\overline{\mathbf{U}}_{a,\mathbf{i}}^{n,n+1}-\Delta t\overline{DF}_{a,\mathbf{i}}^{C}, \tag{3.16}$$

where $\overline{\mathbf{U}}_{a,i}^{n,n+1}$ has the same definition as in 1D algorithm,

$$\overline{\mathbf{U}}_{\mathbf{i}}^{n,n+1}=\mathbf{U}(\mathbf{x}_{\mathbf{i},c}^{n+1},t^n). \tag{3.17}$$

To avoid the small cell problem of the finite volume method, we use a linear combination of conservative and non-conservative flux differences

$$\overline{\mathbf{U}}_{a,\mathbf{i}}^{n+1}=\overline{\mathbf{U}}_{a,\mathbf{i}}^{n,n+1}-\Delta t\left[\kappa_{\mathbf{i}}\,\overline{DF}_{a,i}^{C}+(1-\kappa_{\mathbf{i}})\,\overline{DF}_{a,i}^{NC}\right]. \tag{3.18}$$

Therefore the mass difference between conservative update method (3.16) and non-conservative but stable update method (3.18) is

$$\delta M_{a,\mathbf{i}} = \kappa_{a,\mathbf{i}}^{n+1} \left[\overline{DF}_{a,\mathbf{i}}^{C} - \left(\kappa_{a,\mathbf{i}}^{n+1} \overline{DF}_{a,\mathbf{i}}^{C} + \left(1 - \kappa_{a,\mathbf{i}}^{n+1} \right) \overline{DF}_{a,\mathbf{i}}^{NC} \right) \right] \Delta t. \tag{3.19}$$

Then the mass difference is distributed to the neighbor cells of cell **i**.

3.3. Algorithm - 2D

At the beginning of each time step we assume that following data are available.

- Front position $\mathcal{F}(n)$ and corresponding level set ψ^n at time step n.
- Geometrical features: volume and interface centroids, volume and area fractions and normal vectors to the faces and interface.
- Conserved variable $\overline{\mathbf{U}}_{a,\mathbf{i}}^{n}$ for each phase evaluated at volume centroid.

The goal is to find the second-order estimate of the interface position, $\mathcal{F}(n+1)$, and conserved variables, $\overline{\mathbf{U}}_{a,\mathbf{i}}^{n+1}$, at time step $n+1$. Algorithm for one time step is as follows:

(1) For each phase, derive cell centered variables from cell centroid data and convert to primitive variables $\mathbf{W}_{\mathbf{i}}^{n}$. Similar to 1D case, linear extrapolation is sufficient. See 3.3.1 for details.

(2) Extrapolate $\mathbf{W}_{\mathbf{i}f}^{n}$ for each phase to the front centroids. Slopes in space are needed for this calculation. See 3.3.2 for details of slope calculation.

(3) Solve 1D Riemann problem in front-normal direction. This procedure includes rotating the system of equations on the front to the front-normal direction and solving the Riemann problem in that direction. See [**23**, §5.3] for details.

(4) Use the v_c^f state of the Riemann solution to propagate the level set, determining $\tilde{\psi}^{n+1}$ to first order. See 3.3.3 for details.

(5) Find interface centroids at time $n+1$ from the estimate $\tilde{\psi}^{n+1}$ using the geometry algorithm of [**21**].

(6) Predictor values for each phase:

   (a) If a cell may become *uncovered*, determine its value $\mathbf{W}_{\mathbf{i}}^{n}$ by extrapolation. See 3.3.4 for details.

(b) For all cells in which $\mathbf{W}_\mathbf{i}^n$ is now known, find high-order slopes. Use symmetric fourth-order stencils where possible, and one-sided stencils otherwise. Apply van Leer limiting. See 3.3.2 for details.

(c) For all cells in which $\mathbf{W}_\mathbf{i}^n$ is known, use upwind-filtered characteristic tracing to find edge states $\mathbf{W}_{\mathbf{i},\pm,d}^n$. This procedure is described in detail in [**12**].

(d) If a given edge has only one edge state, determine the missing edge state by extrapolation. See 3.3.5 for details.

(e) Include transverse flux terms (corner coupling) by method of [**12, 27**]. For irregular domains, this is modified as described in [**14**]. Specifically, if an edge is interior ($\mathbf{W}_\mathbf{i}^n$ is known on both sides of the edge), then enough data exists to apply corner coupling correction prescription of [**12, 27**]. However, exterior edges do not possess enough support data. Each step of the algorithm will therefore consist of two parts: (i) normal calculation on interior edges, (ii) reconstruction of exterior edges with the extrapolation algorithm 3.3.5.

(f) Correct edge states on opposite side of a front for effect of front. See 3.3.6 for details.

(g) Compute non-conservative (stable) flux difference $\overline{DF}_\mathbf{i}^{NC}$, cell centered, from (3.15).

(h) A non-conservative estimate of the time-$(n+1)$ state can be made from

$$\tilde{\mathbf{U}}_\mathbf{i}^{n+1} = \mathbf{U}_\mathbf{i}^n - \Delta t\, DF_\mathbf{i}^{NC} \tag{3.20}$$

(7) Extrapolate $\tilde{\mathbf{W}}_\mathbf{i}^{n+1}$ to the time-$(n+1)$ interface centroids, solve the 1D Riemann problem, average $\tilde{v}_c^f$ and v_c^f to make a second-order accurate front reconstruction.

(8) Solve the $D+1$ – dimensional geometry problem to get space-time centroids and new estimates of the $n+1$ volume and boundary centroids using algorithm of [**21**]

(9) For each interface space-time centroid, find left and right states using space-time extrapolation, and solve Riemann problem. See 3.3.7 for details.

(10) For each phase

(a) Extrapolate cell-centered non-conservative flux difference to volume centroids, $\overline{DF}_\mathbf{i}^{NC}$.

(b) Use space-time interpolation to find the states at centroids of edges.

(c) Find the conservative (unstable) flux difference $\overline{DF}_{\mathbf{i}}^{C}$.

(d) Find state $\overline{\mathbf{U}}_{a,\mathbf{i}}^{n+1}$ with a stable hybrid of conservative and non-conservative flux differences.

(e) Compute the conservative mass deficit and redistribute. See 3.3.8 for details.

3.3.1. Cell centered and cell centroid variables. In the beginning of the simulation loop, centroid data for each cell and each phase is available. However, the update algorithm needs cell centered data to evaluate the slopes and fluxes. Therefore a linear extrapolation is used to perform this transfer. That is for the cell $\mathbf{j}$ we fit following lease-square model

$$U_{\mathbf{i}} = U_{\mathbf{j}} + \sum_{d=1}^{D} m_{\mathbf{j},\mathbf{x^d}}(x_{\mathbf{i}}^d - x_{\mathbf{j}}^d) \tag{3.21}$$

where $\mathbf{i}$ is in neighborhood of $\mathbf{j}$, not farther than two cells away and $\kappa_{\mathbf{i}} = 1$ (Figure 3.1).

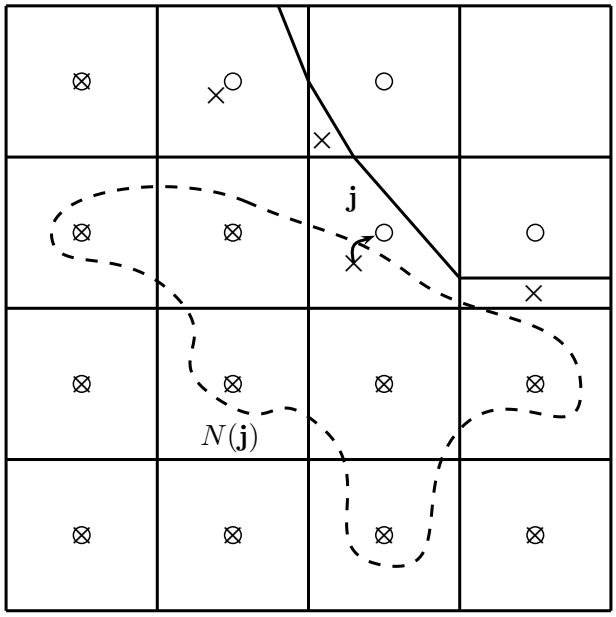

FIGURE 3.1. Extrapolation of cell centroid data to cell centered data using the least-square linear model fitted to the data from neighborhood of the cell.

3.3.2. Slope calculation. To find the slopes we prefer to follow the fourth-order slope calculation from [**14**, §5.1] which uses the fourth-order stencils where possible and one-sided second-order stencils otherwise. Van Leer limiting is also applied to avoid introducing artificial minima and maxima [**28**].

3.3.3. Level set propagation. The level set equation is

$$\psi_t + \mathbf{v}_{ext} \cdot \nabla\psi = 0. \tag{3.22}$$

The extension velocity, $\mathbf{v}_{ext}$, is defined in a way that

$$|\mathbf{v}_{ext}| = v_c^f, \quad \text{on the the front,} \tag{3.23}$$

$$\nabla(\mathbf{v}_{ext,d}) \cdot \nabla\psi = 0, \quad d = 1, \ldots, D. \tag{3.24}$$

Choosing these conditions guarantees that ψ stays as a distance function, $|\nabla\psi| = 1$, under level set equation (3.22) [**1**]. The extension velocity of points on the front may be found by extrapolation between velocity of centroid points that are calculated from Riemann problem. For the points away from the front, we can extrapolate the velocity from the points on the front, knowing that velocity only has normal component to the front [**22**] (procedure is shown in Figure 3.2). Also, we may limit the velocity calculation and level set propagation to the area around the interface, since we only need front position at next time step that is in the close neighborhood of its current position. Therefore, we can update the level set to next time step and find the front position as the zero level set.

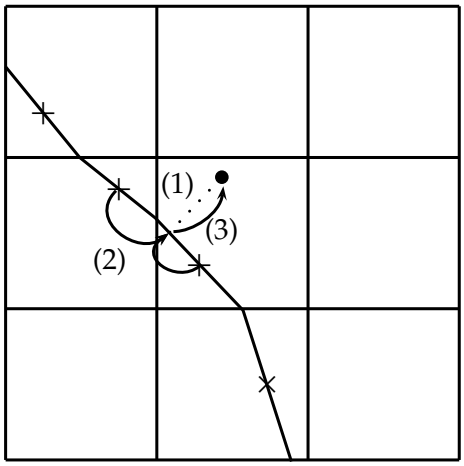

FIGURE 3.2. Extrapolation of external velocity of points off the front (zero level set). (1) Find the normal ray from the external point to the front. (2) Interpolate the velocity to the intersection point from closest front centroids. (3) Extrapolate velocity to the external point.

3.3.4. Newly uncovered cells. As the front moves from one cell to a covered cell, we need to calculate an estimate of variables in the newly uncovered cell. This may be

done by extrapolating data from the neighbor cells. In the extrapolation process we may use cell centered values and corresponding slopes (see 3.3.5 below for such method) or only cell center values. For the second method, we find the line passing from the center of newly uncovered cells normal to the front. Then we find two intersections of the line with grids made from the centers of the neighbor cells and extrapolate from the centers of the neighbor cells to those intersections. Using data on those two points we may extrapolate back to the center of the newly uncovered cell (Figure 3.3).

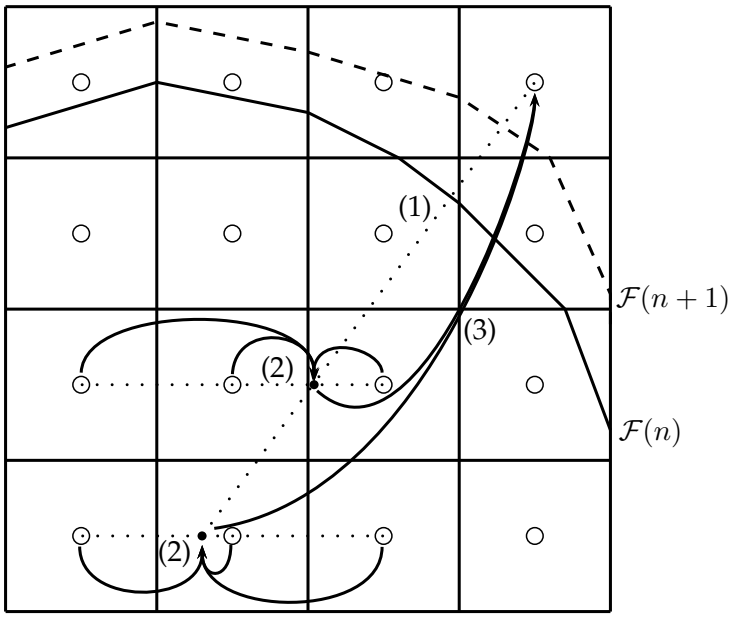

FIGURE 3.3. Extrapolation to newly uncovered cell. (1) Find the ray from the center of newly uncovered cell normal to front. (2) Interpolate data from center of regular cells to the intersection points. (3) Extrapolate back to the center of uncovered cell along the ray.

3.3.5. Extrapolation to covered faces. For a covered face, the data from one side of the face is not available to be used as an initial value for the Riemann problem on that face. To find that value we extrapolate data along the ray that comes from the center of the face normal to the front using local slope values and interpolated data from the same side of the faces of the neighbor cells [**14**, §5.2]. The only restriction here is the cell that is used for data interpolation should not have a covered face. The procedure is shown in Figure 3.4.

3.3.6. The jump condition on the front. The variables that are used to find the flux on other side of front should be consider for the front jump condition. Similar to the 1D case, we extrapolate variables on such faces to the front in the normal direction using the slopes from each phase for the corresponding sides of the front. Then we solve the Riemann problem by rotating the dynamics to the front normal direction. Last, we extrapolate back to the face using slopes from the phase side that the face is defined covered or irregular.

3.3.7. Extrapolation to the front time-space centroid. After evaluating the variation of the front in time and calculating the time centroids of the aperture, we need to

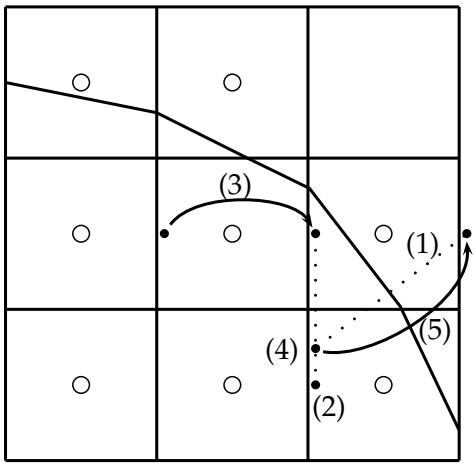

FIGURE 3.4. Extrapolation to covered faces. (1) Find the ray from the covered face normal to the front and find the intersection of the ray and grids with the same direction as the covered face. (2) Find the nearest face states with the same direction as the state of the covered face. (3) If one of the face states is from a cell with a covered face, extrapolate the corresponding data from the nearest cell using the local slope. (4) Interpolate face states to the intersection point. (5) Extrapolate the face state back to the covered face using local slopes.

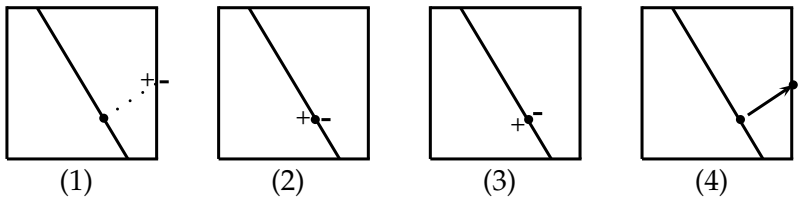

FIGURE 3.5. Implying the jump condition on the front for the states of a covered or irregular face. (1) Find the ray from the face normal to the front. (2) Extrapolate the left and right states to the front using the slopes from the each phase. (3) Solve the Riemann problem in the normal direction to the front. (4) Extrapolate back the data to the face using slope from side which state is going to be used for.

find the fluxes on the aperture and face centroids. For the face centroids that are located on the Cartesian grid we find the variables by extrapolating edge centered fluxes at time n and $n+1$. For the aperture centroids, similar to the 1D case, the cell centered values are extrapolated in time and space to the aperture centroids from each phase. Then the Riemann problem is solved in the normal direction and the flux is calculated. We use the limited slopes and non-conservative flux difference for time-space extrapolation to the aperture centroids,

$$W_{a,\mathbf{i}}^{n_c} = W_{a,\mathbf{i}}^{n} + \sum_{d=1}^{D} \left(\delta x_d^f \, \Delta_{a,4}^d W_{a,i}^d \right) \mp \bar{\alpha}_{a,i\pm\frac{1}{2}} \Delta t \, \Delta_U W \left(\overline{DF}_{a,\mathbf{i}}^{NC} \right), \tag{3.25}$$

where δx_d^f is the position difference between center of the cell and aperture centroid in dth Cartesian direction.

3.3.8. Mass difference redistribution. The mass difference between hybrid methods and conservative method (3.19) should be redistributed to achieve overall conservation. In general, the redistribution scheme is

$$\overline{\mathbf{U}}_{a,\mathbf{i}'}^{n+1} := \overline{\mathbf{U}}_{a,\mathbf{i}'}^{n+1} + w_{\mathbf{i},\mathbf{i}'} \, \delta M_{a,\mathbf{i}}, \quad \mathbf{i} \in N(\mathbf{i}'), \tag{3.26}$$

where $N(\mathbf{i})$ indicates a set of cells in neighborhood of cell $\mathbf{i}$ and weight coefficients are satisfying following conditions [**14**],

$$w_{\mathbf{i},\mathbf{i}'} \geq 0, \qquad \sum_{\mathbf{i}' \in N(\mathbf{i})} w_{\mathbf{i},\mathbf{i}'} \, \kappa_{\mathbf{i}'} = 1. \tag{3.27}$$

We may use an approach similar to the 1D case by defining the acoustic impedance on each side of the front, (2.53) and (2.54), and taking into account the reflection and refraction of waves in front-normal direction, similar to (2.55) and (2.56). Therefore the weight coefficient is defined as follows,

$$w_{\mathbf{i},\mathbf{i}'} = \frac{\kappa_{\mathbf{i}'}}{\sum_{\mathbf{i}'' \in N(\mathbf{i})} \kappa_{\mathbf{i}''}}, \tag{3.28}$$

where here $N(\mathbf{i})$ is the neighborhood of cell $\mathbf{i}$ containing itself for each phase.

Also, in appropriate dynamics, we may apply the scheme from [**14**] which was tested for problems in gas dynamics involving strong shock waves. That scheme uses the density in each cell to produce the weight coefficients.

CHAPTER 4

Summary and future works

In this report we tried to achieve an overall understanding of the methods in the field of moving interfaces. Different approaches have their own benefits and problems. Based on design criteria we chose to use the finite volume method on a Cartesian grid. To understand the basics of this method the 1D algorithm from [**17**] is studied in detail, which brought in the importance of considering correct geometrical features in the method and accurate interpolation/extrapolation schemes to achieve a high order general accuracy in the solution. Having these in mind we proposed a finite volume front-tracking method in 2D. Adding one spatial dimension to the method introduced some changes in algorithm such as the scheme to handle the interface update, in which we applied the level set method, and interpolation and extrapolation methods in which we incorporated the methods from [**14**].

The next step is to implement the 1D and 2D method. Although the implementation of the 1D method [**17**] showed promising results, we need to try it ourselves. The details of realization in 1D will be helpful for implementing the 2D method. We also mentioned choices for some parts of the 2D algorithms. We have to find out which one performs better in different cases and do the needed adjustments to reach a desired performance. One choice for the implementation infrastructure is the Chombo package [**2**] which provide essentials for grid generation and indexing and makes easier to add extension to higher dimension or adaptive mesh refinement algorithms.

Bibliography

[1] Adalsteinsson, D., and Sethian, J. A. The fast construction of extension velocities in level set methods. *Journal of Computational Physics 148*, 1 (1999), 2 – 22.

[2] Applied Numerical Algorithms Group of Lawrence Berkeley National Lab. Chombo. `https://seesar.lbl.gov/ANAG/chombo/`.

[3] Bell, J. B., , Colella, P., and Welcome, M. L. Conservative front-tracking for inviscid compressible flow. In *American Institute of Aeronautics and Astronautics Conference* (1991), A. D. Vakili and C. Gauthier, Eds., pp. 24–26.

[4] Belytschko, T., and Black, T. Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering 45*, 5 (1999), 601 – 620.

[5] Berger, M. J., Helzel, C., and Leveque, R. J. h -box methods for the approximation of hyperbolic conservation laws on irregular grids. *SIAM Journal on Numerical Analysis 41*, 3 (2003), 893.

[6] Berger, M. J., and LeVeque, R. J. Stable boundary conditions for cartesian grid calculations. *Computing Systems in Engineering 1*, 2 (1990), 305 – 311.

[7] Chern, I.-L., and Colella, P. A conservative front tracking method for hyperbolic conservation laws. *Lawrence Livermore National Laboratory LLNL Report No. UCRL-97200* (1987).

[8] Chessa, J., and Belytschko, T. An enriched finite element method and level sets for axisymmetric two-phase flow with surface tension. *International Journal for Numerical Methods in Engineering 58*, 13 (2003).

[9] Chessa, J., and Belytschko, T. An extended finite element method for two-phase fluids. *Journal of Applied Mechanics 70*, 1 (2003), 10–17.

[10] Chorin, A. J. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation 22*, 104 (1968), 745–762.

[11] Cohen, I. M., and Kundu, P. K. *Fluid Mechanics, Third Edition*. Academic Press, 2004.

[12] Colella, P. Multidimensional upwind methods for hyperbolic conservation laws. *Journal of Computational Physics 87*, 1 (1990), 171 – 200.

[13] Colella, P., Glaz, H. M., and Ferguson, R. E. Multifluid algorithms for eulerian finite difference methods, 1996. manuscript.

[14] Colella, P., Graves, D. T., Keen, B. J., and Modiano, D. A cartesian grid embedded boundary method for hyperbolic conservation laws. *Journal of Computational Physics 211*, 1 (2006), 347 – 366.

[15] Du, J., Fix, B., Glimm, J., Jia, X., Li, X., Li, Y., and Wu, L. A simple package for front tracking. *Journal of Computational Physics 213*, 2 (2006), 613 – 628.

[16] Flanders, H. Differentiation under the integral sign. *The American Mathematical Monthly 80*, 6 (1973), 615–627.

[17] Gatti-Bono, C., Colella, P., and Trebotich, D. A second-order accurate conservative front-tracking method in one dimension. Unpublished.

[18] Gerstenberger, A., and Wall, W. A. An extended finite element method/Lagrange multiplier based approach for fluid-structure interaction. *Computer Methods in Applied Mechanics and Engineering 197*, 19-20 (2008), 1699 – 1714. Computational Methods in Fluid-Structure Interaction.

[19] Huerta, A., and Liu, W. K. Viscous flow with large free surface motion. *Computer Methods in Applied Mechanics and Engineering 69*, 3 (1988), 277 – 324.

[20] LeVeque, R. J. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[21] Ligocki, T. J., Schwartz, P. O., Percelay, J., and Colella, P. Embedded boundary grid generation using the divergence theorem, implicit functions, and constructive solid geometry. *Journal of Physics: Conference Series 125* (2008), 012080 (5pp).

[22] Malladi, R., Sethian, J. A., and Vemuri, B. C. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17* (1995), 158–175.

[23] Miller, G. H., and Colella, P. A conservative three-dimensional eulerian method for coupled solid-fluid shock capturing. *Journal of Computational Physics 183*, 1 (2002), 26 – 82.

[24] Miller, G. H., and Puckett, E. G. A high-order Godunov method for multiple condensed phases. *Journal of Computational Physics 128*, 1 (1996), 134 – 164.

[25] Ortiz, M., and Pandolfi, A. Finite-deformation irreversible cohesive elements for three-dimensional crack-propagation analysis. *International Journal for Numerical Methods in Engineering 44*, 9 (1999), 1267 – 1282.

[26] Peskin, C. S. The immersed boundary method. *Acta Numerica 11* (2002), 479 – 517.

[27] Saltzman, J. An unsplit 3d upwind method for hyperbolic conservation laws. *Journal of Computational Physics 115*, 1 (1994), 153 – 168.

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