UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

VINICIUS DA COSTA AZEVEDO

Efficient Smoke Simulation on Curvilinear Grids

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Manuel Menezes de Oliveira Neto
Advisor

Porto Alegre, December 2012

CIP – CATALOGING-IN-PUBLICATION

Azevedo, Vinicius da Costa

Efficient Smoke Simulation on Curvilinear Grids / Vinicius da Costa Azevedo. – Porto Alegre: PPGC da UFRGS, 2012.

71 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2012. Advisor: Manuel Menezes de Oliveira Neto.

1. Three-Dimensional Graphics and Realism - Animation. 2. Types of Simulation - Animation. 3. Fluid simulation. 4. Curvilinear grids. 5. Overlapping grids. I. Oliveira Neto, Manuel Menezes de. II. Title.

“Do, or do not. There is no ’try’.”
— YODA (’THE EMPIRE STRIKES BACK’)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This thesis present an efficient approach for performing smoke simulation on curvilinear grids. The solution of the Navier-Stokes equations on curvilinear is made on three steps: advection, pressure solving and velocity projection. The proposed advection method is simple, fast and unconditionally-stable. Our solution is able to maintain a staggered-grid variable arrangement, and includes an efficient solution to enforce mass conservation.

Compared to approaches based on regular grids traditionally used in computer graphics, our method allows for better representation of boundary conditions, lending to more realistic results, with just a small increment in computational cost. Moreover, we are able to condensate cells where interesting artifacts tend to appear, like swirling vortices or turbulence. We demonstrate the effectiveness of our approach, both in 2-D and 3-D, through a variety of high-quality smoke simulations and animations. These examples show the integration of our method with overlapping grids and multigrid techniques.

Keywords: Three-Dimensional Graphics and Realism - Animation, Types of Simulation - Animation, Fluid simulation, Curvilinear grids, Overlapping grids.

1 INTRODUCTION

Fluid animation tries to visually mimic the complex behavior of fluids in motion such as water, smoke, and explosions. These natural events are pervasive in a wide range of applications in computer graphics. Thus, it is important to reproduce plausible representations of these phenomena within virtual environments. Over the years, various successful algorithms have been proposed (Figure 1.1), with the notable presence in films (FEDKIW; STAM; JENSEN, 2001; ENRIGHT; MARSCHNER; FEDKIW, 2002; RASMUSSEN et al., 2004; MOLEMAKER et al., 2008), animations (CHENTANEZ et al., 2007; HONG; SHINAR; FEDKIW, 2007), and interactive games (COHEN; TARIQ; GREEN, 2010; CHENTANEZ; MULLER, 2011a).

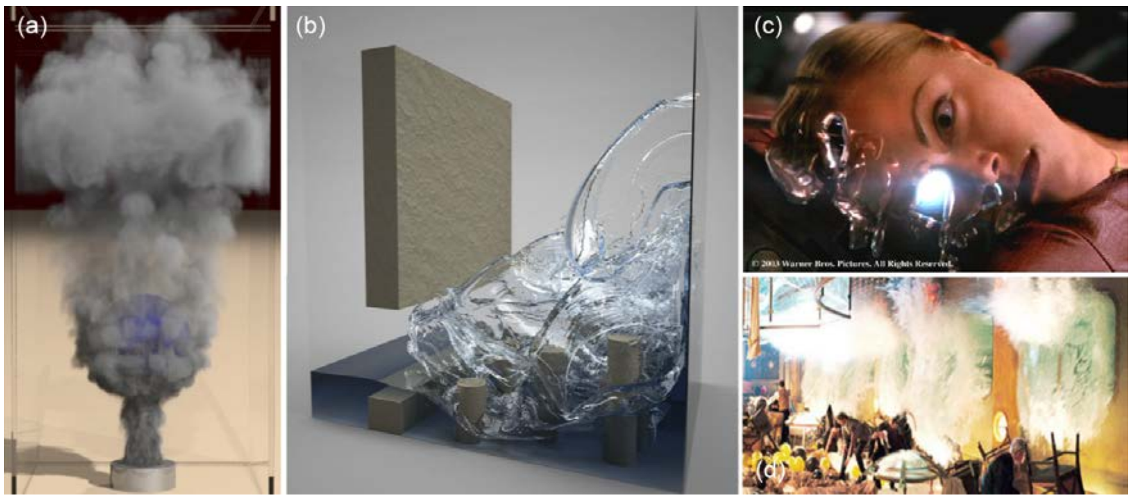


Figure 1.1: Fluid simulation is widely used in animations, films and games: (a) Smoke simulation; (b) Breaking Dam simulation; (c) Scene from Terminator 3; (d) Scene from Poseidon.

Despite the existence of many solutions proposed in the last 15 years, faster and more precise algorithms are needed to meet industry requirements for better and more realistic representations of fluids. For instance, recently a 100-member team of software developers, engineers, and artists were hired for a year to develop specific applications for animating complex fluid phenomena for the film Poseidon (PETERSEN, 2006).

Given a scene configuration and the nature of the fluid flow (e.g., smoke or water), a simulation of the underlying physical process is used to obtain a consistent and plausible fluid representation. The simulation makes use of the Navier-Stokes equations, which are discretizated in both time and space.

In computer graphics, there are two popular approaches for fluid representation: **Lagrangian** (Figure 1.2) and **Eulerian** (Figure 1.3) methods. Lagrangian methods evaluate each blob of fluid separately, representing the entire fluid as a particle system. Eulerian methods, on the other hand, use grids of fixed points distributed on the simulation domain to evaluate the fluid properties. Eulerian methods produced the most impressive visual results to date.

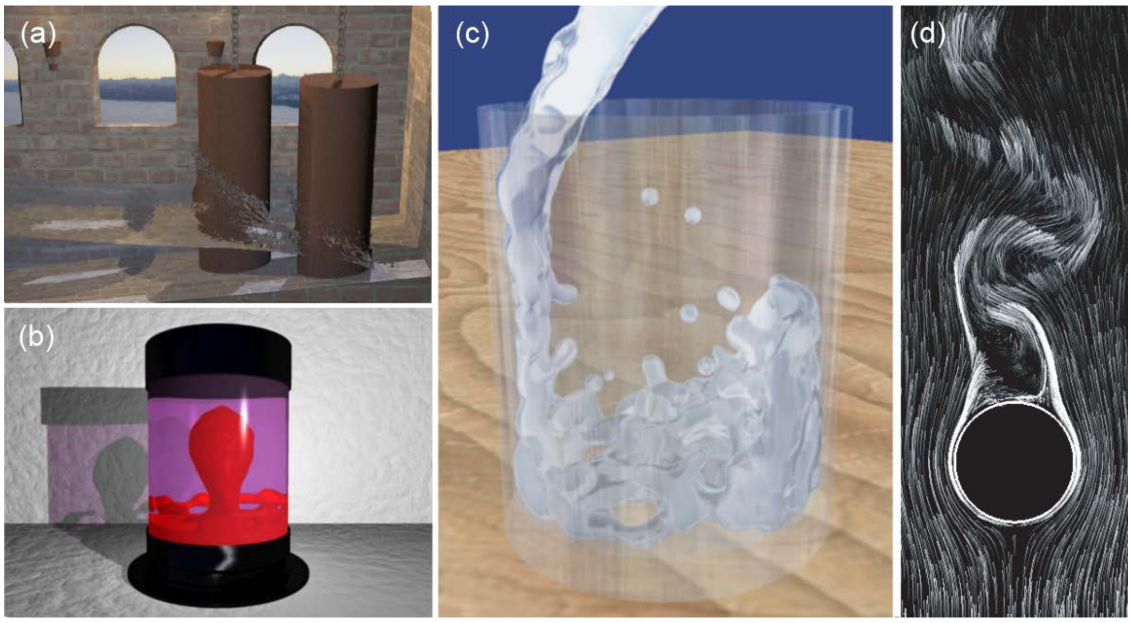


Figure 1.2: Lagrangian examples of fluid discretization: (a) Water colliding with cylinder obstacles; (b) Simulation of lava lamp; (c) Water pouring in glass; (d) Visual appearance of particle-based flow around circular object.

The quality of Eulerian flow simulations depends on two key factors: the specification of *precise boundary conditions*; and *fine resolution in high-vorticity regions of the simulation domain*. The use of regularly-structured simulation grids, as traditionally used in computer graphics (FOSTER; METAXAS, 1997; STAM, 1999; FEDKIW; STAM; JENSEN, 2001), simplifies the simulation process at the expense of using poorly defined boundary conditions. Unstructured grids (FELDMAN; O'BRIEN; KLINGNER, 2005; KLINGNER et al., 2006; CHENTANEZ et al., 2007), on the other hand, can precisely represent boundary conditions. However, since the solver's accuracy depends on the meshes not having stretched or twisted simplices, the resulting grids tend to be highly isotropic. As a result, the need of local refinements to capture vortices often results in grids that are globally much finer than necessary (WYMAN, 2001).

This thesis presents an efficient approach for creating high-quality flow simulations based on *curvilinear grids* – a space discretization commonly used in computational fluid dynamics (CFD) (CHESSHIRE; HENSHAW, 1990). Such grids, also known as *non-regular structured grids*, adapt themselves to the shapes of the objects in the scene, defining precise boundary conditions. As opposed to unstructured grids, curvilinear ones maintain a fixed topology. This simplifies the solution of linear systems, and makes the cost of the flow solver nearly identical to the ones used with regular grids. Moreover, grid cells can be easily refined near obstacles and in high-vorticity areas.

For graphics applications, fluid simulation needs to be both stable and fast. To meet these requirements, we adapt existent curvilinear grid methods from CFD. For the mo-

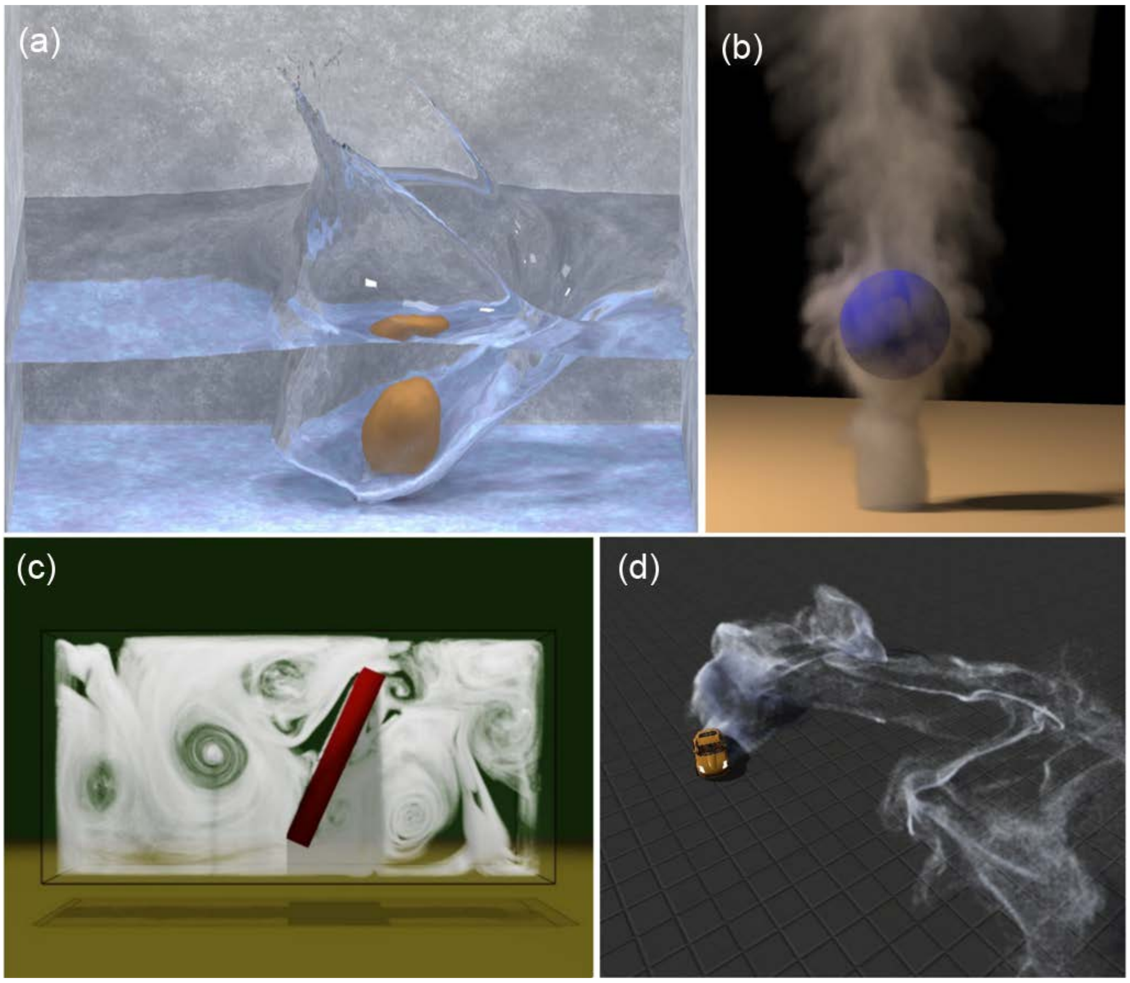


Figure 1.3: Eulerian examples of fluid discretization. (a) Ball splashing water in tank; (b) Smoke around spherical obstacle; (c) Smoke mixed by a paddle in closed tank; (d) Real-time simulation of smoke around moving car.

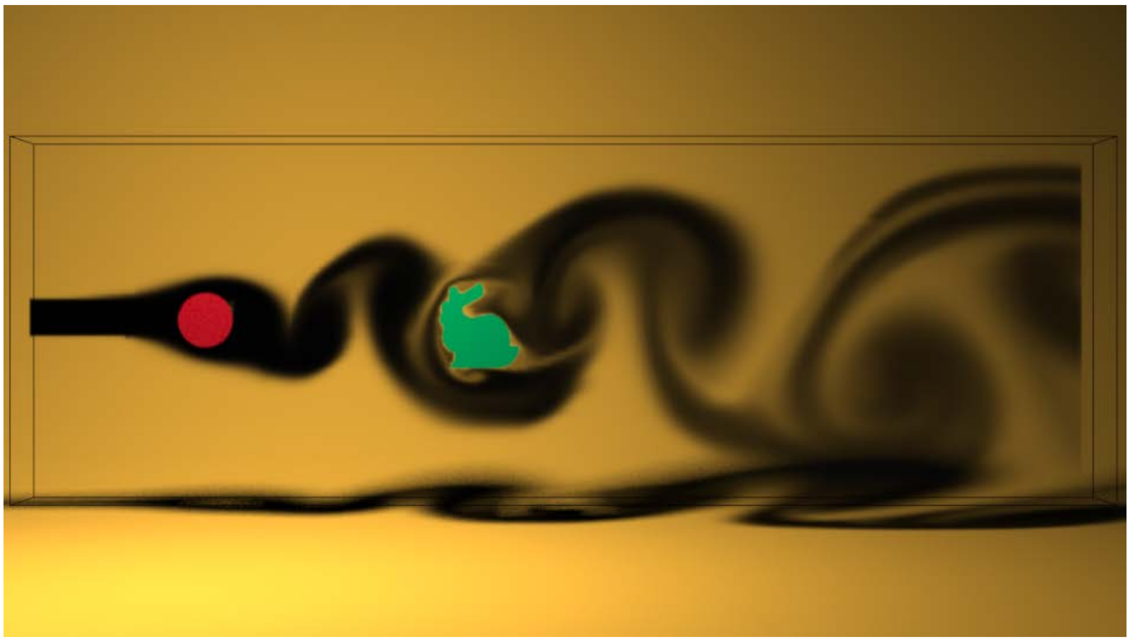
mentum equations, we use a fast semi-Lagrangian method based on a domain transformation scheme. We are also able to maintain the Cartesian staggered arrangement by introducing a simple and robust mass-conservation method that produces oscillation-free velocity fields.

The use of curvilinear grids, with support for local refinements, guarantees high-quality simulation results. Figure 1.4 shows examples of flow simulations created with our approach. On the left (Figure 1.4a), a 2-D simulation on a channel with multiple obstacles shows that proper boundary conditions can be maintained, even for complex shapes. The image on the right (Figure 1.4b) shows a 3-D smoke simulation on a wind tunnel. These examples illustrate the potential and flexibility of our approach.

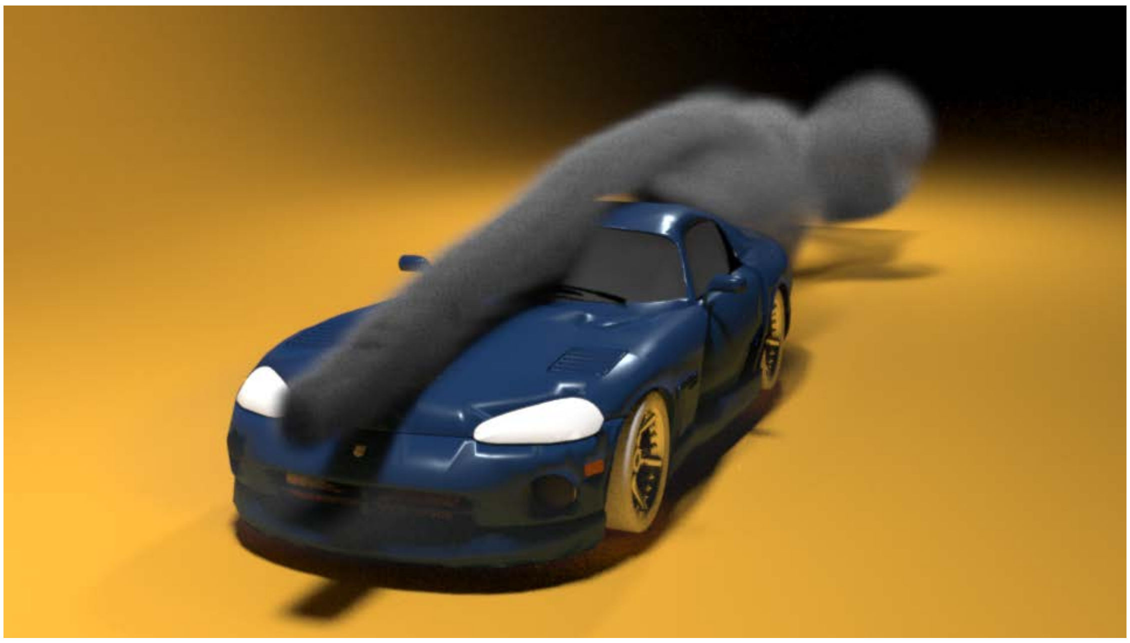
The **contributions** of this thesis include:

- *An efficient approach for creating high-quality flow simulations on curvilinear grids* (Chapter 3). Our technique allows for precise definition of boundary conditions, leading to more realistic simulations and animations than regular-grid-based approaches. It is also significantly faster than unstructured-grid-based techniques, while producing results of similar quality;

- *A new technique to enforce mass conservation* (Section 3.3) that works with a



(a)



(b)

Figure 1.4: Examples of smoke simulations produced with our technique. (left) 2-D simulation on a channel with multiple obstacles. Note the boundary conditions properly defined, and the vortices formed behind the obstacles. The simulation plane is inside a box, and a light source casts shadows of the smoke on the floor. (right) 3-D smoke simulation in a wind tunnel to visually evaluate a car's aerodynamics properties.

staggered-grid variable arrangement. Our solution is faster than the traditional one used in CFD, based on D'Yakunov's method (CONCUS; GOLUB, 1972).

- *The introduction to computer graphics graphics of a fast CFD unconditionally-stable advection algorithm for curvilinear grids* (Section 3.1). The algorithm uses adaptive time steps to generate more accurate results.

1.1 Thesis Structure

This thesis is structured as follows: Chapter 2 discusses some of the fluid animation techniques and methods in computer graphics. Chapter 3 presents our approach for solving the Navier-Stokes equations on curvilinear overlapping domains. Chapter 4 discusses our multigrid technique and grid generation methods. Chapter 5 shows our results, comparing them to traditional regular grid schemes. Chapter 6 summarizes the thesis and discusses some ideas for future work.

2 RELATED WORK

Fluid simulation essentially consists of evaluating the Navier-Stokes equations. In computer graphics, this is done using one of two popular methods: Lagrangian, and Eulerian approaches. In the Lagrangian approach, each blob of fluid is evaluated separately, representing the fluid as a particle system. With the introduction of Smooth Particle Hydrodynamics (SPH) (MULLER; CHARYPAR; GROSS, 2003), Vortex Particles (PARK; KIM, 2005) and hybrid Vortex Particles (SELLE; RASMUSSEN; FEDKIW, 2005), Lagrangian methods became popular in real time applications. The main advantages of this method is that it adapts instantaneously to the environment, scales with asymptotic complexity $O(n \log n)$ and facilitates the extraction of complex fluid surfaces from the particle system. Lagrangian methods tend to be less efficient as they approach the incompressible limit. Thus, they are rarely used for animating smoke. Recent advances in Lagrangian methods include weakly compressible equation-of-state formulations (BECKER; TESCHNER, 2007), predictive-corrective schemes (SOLENTHALER; PAJAROLA, 2009) and ghost SPH methods (SCHECHTER; BRIDSON, 2012).

Eulerian methods use grids of fixed points distributed on the domain to evaluate the fluid properties. Such grids are obtained through successive domain subdivisions (FOSTER; METAXAS, 1997; STAM, 1999; FEDKIW; STAM; JENSEN, 2001; ENRIGHT; MARSCHNER; FEDKIW, 2002). There is a great amount of work on the subject of Eulerian methods. Fedkiw et al. and Foster and Fedkiw presented methods for high quality representations of smoke (FEDKIW; STAM; JENSEN, 2001) and liquids (FOSTER; FEDKIW, 2001). Enright et al. introduced the particle level-set, which increased the accuracy of complex water surfaces (ENRIGHT; MARSCHNER; FEDKIW, 2002). These methods were further extended to handle fire (NGUYEN; FEDKIW; JENSEN, 2002), explosions (FELDMAN; O'BRIEN; ARIKAN, 2003), multiple interactive fluids (LOSASSO et al., 2006), wrinkled flames and cellular patterns (HONG; SHINAR; FEDKIW, 2007) and two-way solid-fluid coupling (ROBINSON-MOSHER et al., 2008).

In the context of improving the baseline simulation, Stam introduced to the computer graphics community the popular stable fluids method (STAM, 1999). Molemaker et al. used the QUICK advection scheme to improve the quality of low-viscosity animations (MOLEMAKER et al., 2008). Selle et al. presented the Modified MacCormack method, a low dissipation unconditionally stable advection scheme (SELLE et al., 2008). Doyub et al. further improved previous works by adapting the high-order CIP method to computer graphics environments (KIM; SONG; KO, 2008).

The usual grid layout used in Eulerian simulations consists of a structured regular subdivision, discretizing the environment in a voxelized fashion (Figure 2.1a). Although regular subdivision is a common and reliable technique, it produces rough representations for object geometry, as shown in Figure 2.1a. The resulting incorrect boundaries introduce

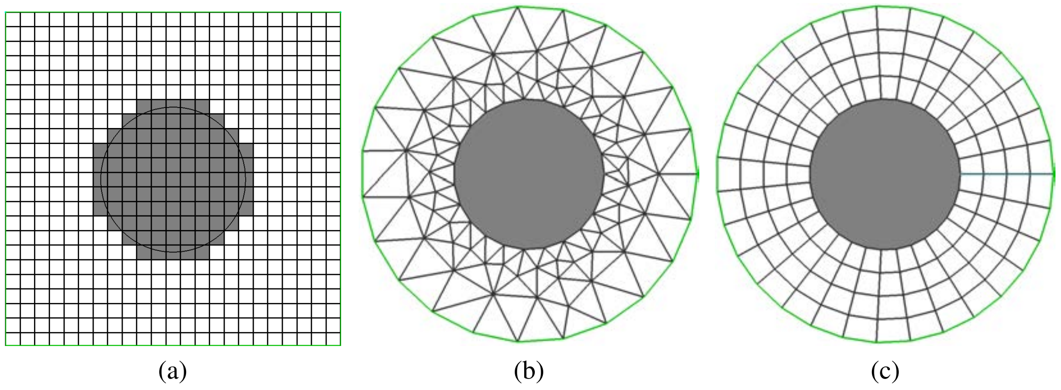


Figure 2.1: Space-discretization techniques: (a) regular grid; (b) unstructured grid; and (c) non-regular structured grid.

noticeable artifacts in the simulations/animations, which do not vanish with increased grid resolution (FELDMAN; O'BRIEN; KLINGNER, 2005; BATTY; BERTAILS; BRIDSON, 2007; ELCOTT et al., 2007; WENDT et al., 2007).

2.1 Precise Boundary Conditions

Some authors have attempted to mitigate the artifacts generated by incorrect boundary specification. Foster and Fedkiw updated tangential velocities using boundary normals information, constraining velocities to not pass through solid walls (FOSTER; FEDKIW, 2001). Houston et al. improved the accuracy of the previous method using a level set to represent objects geometry instead of relying on polygon meshes directly (HOUSTON; BOND; WIEBE, 2003). Rasmussen et al. further elaborated this approach for level set advection (RASMUSSEN et al., 2004). However, all of the presented methods suffer from the voxelized artifacts during the simulation, since they do not modify the pressure system to accommodate complex boundaries.

Batty et al. removed voxelized pressure artifacts using a method based on a kinetic-energy minimization (BATTY; BERTAILS; BRIDSON, 2007). Boundary cells are able to be partially filled by object boundaries. Then, the pressure matrix is changed, allowing the fluid to occupy the partially filled boundary. Robinson-Mosher et al. extended this work to fully two way coupled fluid/solid simulation, supporting thin and small objects (ROBINSON-MOSHER; ENGLISH; FEDKIW, 2009).

None of these methods are able to precisely enforce aerodynamical properties, such as a flow around an airfoil. This happens because the discretization employed in the pressure system does not take into account filled fluid areas which asymmetrically contribute to internal forces of a partially occupied fluid cell. Roble et al. minimizes that by modifying the divergent approximation on boundary cells, calculating it using a finite volume formulation (ROBLE; ZAFAR; FALT, 2005). Mapping and tracking the modified boundary cells in this method presents a common implementation difficulty, specially for three-dimensional objects. Moreover, if the grid is not fine enough, the mapped cells normals may poorly represent the actual object boundary.

Moreover, Roble et al. uses a simplified version of the Immersed Boundary Method (IBM) (ROBLE; ZAFAR; FALT, 2005), which has been extensively developed in CFD area. For an extensive review of different Immersed Boundary Methods see (BAN-

DRINGA, 2010).

2.2 Adaptive Grid Refinement

In order to capture small scale details of the flow and to better represent boundary conditions, Losasso et al. optimize the domain cell distribution using octrees (LOSASSO; GIBOU; FEDKIW, 2004). Losasso et al. further improved the method to better handle surfaces on T-junctions (LOSASSO; FEDKIW; OSHER, 2005). Houston et al. introduced the Hierarchical Run-Length Encoded (H-RLE) Level Set data structure, which supports big data sets, exceeding 45 billion voxels (HOUSTON et al., 2006). Doyub et al. combined an adaptive approach to the Eulerian vortex sheet method, creating complex surface details, such as thin and wiggling fluids (KIM; SONG; KO, 2009).

Octree-based approaches can improve the use of computational resources, however its use leads to non-symmetric systems in the pressure equation. To overcome this, the authors simplify the representation of the pressure gradient (LOSASSO; GIBOU; FEDKIW, 2004). This drops the projection step to first order accuracy in space, hindering the overall exactness of the method.

Tall-cell grids (IRVING et al., 2006) combine 2-D and 3-D techniques for simulating large bodies of water. This approach condensates the cells near the water interface, coarsening the grid on regions where the pressure profile is constant. Chentanez and Muller adapted this technique for real-time environments (CHENTANEZ; MULLER, 2011a). Tall-cell grids are only useful in scenarios where large bodies of water are present, such as tanks, rivers and oceans.

2.3 Non-Regular Grids

Non-regular grids consist of domain elements that tightly fit the boundaries of the fluid obstacles (KIM; CHOI, 2000; FELDMAN; O'BRIEN; KLINGNER, 2005; KLINGNER et al., 2006; TILCH et al., 2008). This is the standard approach in CFD due to its accuracy. However, any changes in object shape or in the spatial relationship among objects in the scene require grid regeneration. Non-regular grids can be classified as unstructured or structured.

Unstructured grids often discretize the solution domain using triangles (2-D) (Figure 2.1b) or tetrahedra (3-D). They were introduced to computer graphics by Feldman et al. (FELDMAN; O'BRIEN; KLINGNER, 2005). In their work, the simulation domain is composed by fixed unstructured tetrahedral meshes and regular hexahedral cells, combining accuracy near obstacles and efficiency in open regions. The work was later extended to dynamic environments (FELDMAN et al., 2005; KLINGNER et al., 2006; CHENTANEZ et al., 2007) using a mesh re-generation technique, which can take up to forty percent of the total simulation time (KLINGNER et al., 2006).

The main advantage of unstructured grids is the ability to discretize highly-complex geometries, with acute angles and concavities. Also, grid generation is fast and automated. However, unstructured grids have no discernible organized structure, and node locations and neighbors need to be specified explicitly. For fluid simulation, this implies that more sophisticated and robust algorithms are required to solve the resulting system of equations, causing its solution to be slower than for structured grids. Also, concentrating cells in high-vorticity regions can be a complex task. This happens because triangle and tetrahedral elements do not stretch or twist well without affecting the stability and convergence of the flow solver, limiting the grid mesh to some level of isotropy. Therefore, it is often necessary to refine large portions of the grid to achieve local refinements (WYMAN, 2001).

Non-regular structured grids, also known as *curvilinear grids*, are based on tessellations of an N-dimensional Euclidean space, adaptively contouring objects and filling the simulation domain without gaps (Figure 2.1c). The structured patterns of these grids simplify the evaluation of the associated equations. The grid can be re-shaped by stretching, shearing, or bending without changing its topology. Curvilinear grids are constructed using quadrilaterals (in 2-D) or hexahedra (in 3-D). Each point in the grid has four nearest neighbors in 2-D, and six in 3-D. In order to increase the overall solution accuracy and stability, grid generators use elliptic equations to optimize the shape of the mesh for orthogonality and uniformity (WYMAN, 2001).

Curvilinear grids have been used in computer graphics to produce visual effects on objects textures (STAM, 2003), but the grids were limited to Catmull-Clark surfaces. Stam's approach is unable to generate arbitrary curvilinear grids to precisely represent boundary conditions, and cannot support local cell refinements. Due to differences between arbitrary curvilinear grids and Catmull-Clark surfaces, a straightforward implementation of the method in (STAM, 2003) produced velocity oscillations in our simulations. Moreover, Stam's algorithm does not use a domain transformation in the advection phase. Thus, a local search to find the correct cell index of the backtracked particle is needed, negatively affecting the performance of the algorithm.

Barroso and Celes adapted the approach of (STAM, 1999) to arbitrarily shaped curvilinear grids (BARROSO; CELES, 2011). They use Jacobian matrices to transform the grid into a computational space, thus simplifying the solution of the Navier-Stokes equations. However, their method is based on collocated grid schemes, which does not support strong coupling between velocities and pressures (ZANG; STREET; KOSEFF, 1994). Moreover, Barroso and Celes use a single structured grid to cover all the domain, difficulting the grid generation procedure and limiting the applicability of the technique.

In this thesis, we exploit the desirable properties of curvilinear grids of correctly representing boundary conditions and supporting local refinement to create an efficient flow-simulation approach. For this, we have developed a new velocity projection technique that works with a staggered-grid variable arrangement, use a fast unconditionally-stable advection algorithm and decompose the domain into multiple grids. In order to increase the applicability of our technique, we used a domain decomposition method (FERZIGER; PERIC, 1999; HENSHAW, 2005a). Our technique can produce high-quality simulation results and is considerably faster than approaches based on unstructured grids.

2.4 Variable Arrangement

The common choice of variable arrangement for regular grids is to use the *MAC approach* (HARLOW; WELCH, 1965). By storing different variables at different locations, this staggered-Cartesian-grid scheme (Figure 2.2a) guarantees strong coupling between pressure and velocity. For curvilinear grids, the choice of variable arrangement is often subject to discussion. The main argument against the staggered-Cartesian scheme is that it may not contribute to the flux of a face when the grid lines turn 90 degrees with respect to the orientation of the Cartesian component. Such a situation results in spurious oscillations in the velocity field as the simulation advances.

Many works use different variables arrangement schemes with curvilinear grids. Colocated schemes (ZANG; STREET; KOSEFF, 1994) (Figure 2.2b) store all the variables at the center of cell - regularizing terms are needed to guarantee the strong coupling of momentum and pressure equations. These terms may often falsify transient behavior and make transient flow computations more costly and complicated (WESSELING; SEGAL; KASSELS, 1999). Full-staggered schemes (Figure 2.2c) store all the velocity components midway of pressures variables, but they are much slower since additional variables need to be computed in the momentum equations. Grid-oriented component schemes (Figure 2.2d) transform the velocity according to the grid curvature, but this arrangement introduce non-conservative terms, invalidating the use of weakly-conservative differential momentum forms (Eq. (3.1)).

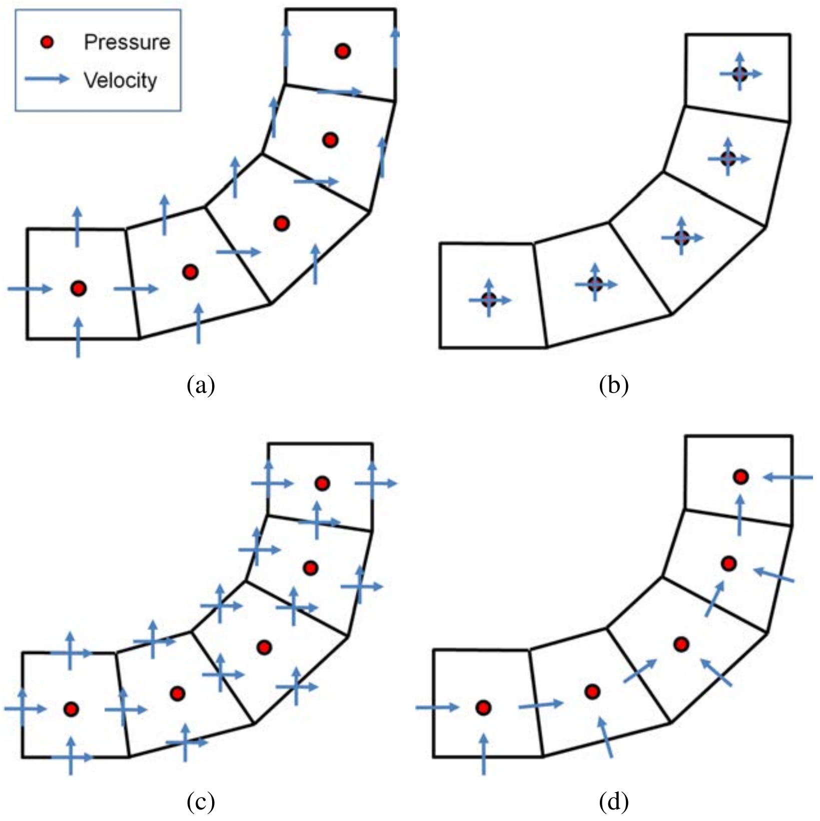


Figure 2.2: Velocity arrangements in staggered grids: (a) Staggered Cartesian. (b) Collocated Cartesian. (c) Full-staggered Cartesian. (d) Staggered grid-oriented.

As noted in (SHYY; VU, 1991), there is no problem when using staggered Cartesian oriented velocities as long two conditions do not hold at the same time: grid turns of exact 90 degrees with respect Cartesian components **and** constant metric terms (no variation in grid cell spacing). This is a rare case scenario for structured grids, an this scheme was adopted to preserve the overall method simplicity. Moreover, since our solution uses pressure derivatives for each Cartesian component (Section 3.2), we can safely exploit the computational efficiency of the staggered Cartesian arrangement.

3 EFFICIENT SMOKE SIMULATION ON CURVILINEAR GRIDS

We use the differential form of the inviscid, incompressible Navier-Stokes equations, which are written as

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{f} \tag{3.1}$$

and

$$\nabla \cdot \vec{u} = 0, \tag{3.2}$$

where $\vec{u}$ and p are the velocity and pressure fields, respectively, ρ is the fluid density, and $\vec{f}$ represents additional forces acting on the fluid. Since we are simulating smoke, we decided to drop the viscosity term. This is a reasonable and common assumption that has been adopted by other authors (FEDKIW; STAM; JENSEN, 2001; MOLEMAKER et al., 2008; LENTINE; ZHENG; FEDKIW, 2010). Moreover, advection methods usually present some level of dissipation, which may further be re-interpreted as viscosity. Equations (3.1) and (3.2) are called the *momentum conservation* and the *mass conservation* equations.

The traditional way for solving Eqs. (3.1) and (3.2) is using the *projection method* (CHORIN, 1968), which consists of three main phases: (i) Advection, (ii) Pressure solving; and (iii) Velocity projection. The *advection phase* consists in estimating an intermediate value $\vec{u}^{*(n+1)}$ for the velocity field $\vec{u}$, at time $t^{(n+1)}$, based on its value $\vec{u}^{(n)}$ in the previous time step and computing external forces acting on the fluid (*e.g.*, gravity). *Pressure solving* then computes a scalar field of pseudo pressure values $p^{(n+1)}$ at time $t^{(n+1)}$ that guarantees mass conservation (*i.e.*, enforces that $\nabla \cdot \vec{u} = 0$). Finally, the *velocity projection phase* couples the two equations by computing $\vec{u}^{(n+1)}$, the actual value of the velocity field at time $t^{(n+1)}$, from $\vec{u}^{*(n+1)}$ and $p^{(n+1)}$. Sub-sections 3.1 to 3.3 present the details of our projection method for curvilinear grids.

3.1 Semi-Lagrangian Advection in Curvilinear Grids

The *semi-Lagrangian method* (HOLLY; PREISSMAN, 1977; ROBERT, 1981) is a classic algorithm used in computer graphics (STAM, 1999) for the advection phase in fluid simulation. It is fast, unconditionally stable, and works well for regular domains. The method is formulated by calculating the back trajectory of a particle q from a given point x:

$$\vec{u}^{*(n+1)}(x) = \vec{u}^{(n)}(x_p), \tag{3.3}$$

where

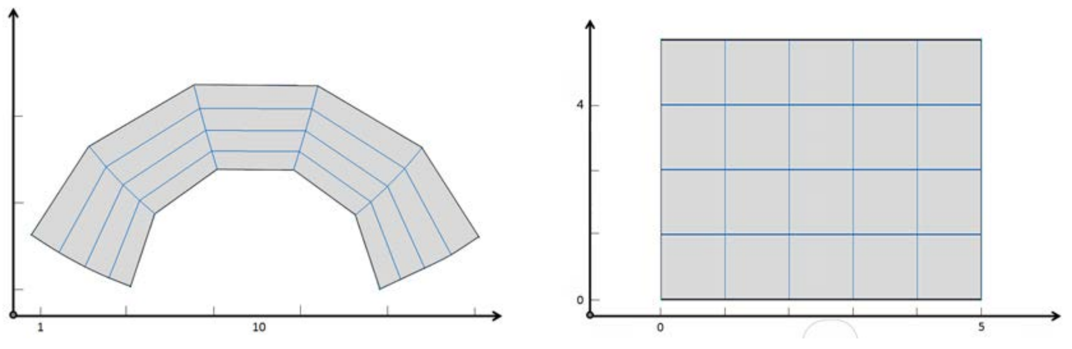
$$x_p = x - \vec{v}^{(n)}(x)\Delta t, \tag{3.4}$$

and $\vec{v}^{(n)}(x)$ is the velocity of particle q at position x and time $t^{(n)}$. $\vec{v}^{(n)}(x)$ is interpolated from the values of $\vec{u}^{(n)}$ stored in the Eulerian grid.

While the semi-Lagrangian method is straightforward in regular grids, it is not directly applicable to non-regular grids. Efficiently identifying the cells containing the backtracked positions x_p is not trivial, as scales may vary arbitrarily among cells, and the grid's curvature may also change. A simple and intuitive solution to this problem would check in which cells the backtracked positions fall, compare the backtracked positions against cell faces, incrementally searching for the correct cell index in the local neighborhood (STAM, 2003). Given the non-regular domain subdivision, this would lead to an overhead in the advection phase, specially in massive parallel architectures where conditional branches should be preferably avoided. Moreover, the velocity interpolation process is also more complex in irregular cells.

CFD researchers (KARPIK; CROCKETT, 1997) transform the (*physical-domain*) non-regular grid Ω onto a (*computational-domain*) canonical regular grid Ω' (Figure 3.1) to simplify the advection of densities through a curvilinear grid. The back trajectories are computed on the canonical grid. The transformations mapping these two domains are similar to model-deformation techniques used in computer graphics (BARR, 1984). Intuitively, they map each cell (a quadrilateral in Figure 3.1 (left)) from the original mesh to a canonical cell (a unit square in Figure 3.1 (right)). Note that these transformations are homeomorphisms and vary from cell to cell.

We extend the approach of Karpik and Crockett to dynamically update the intermediary velocity field using this domain transformation technique. Thus, we are able to achieve the same simplicity, stability, and speed of the standard semi-Lagrangian method traditionally used on regular grids.



(a) (b)

Figure 3.1: Physical domain Ω (a) and canonical computational domain Ω' (b).

The velocity vector $\vec{u}$ can be expressed in the Cartesian frame of reference as

$$\vec{u} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = \vec{u}_{ijk}, \tag{3.5}$$

where $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ are canonical unit vectors in the X, Y, and Z axis directions. Alternately, $\vec{u}$ can be represented as

$$\vec{u} = u_1\mathbf{g^1} + u_2\mathbf{g^2} + u_3\mathbf{g^3} = \vec{u}_{\xi\eta\tau}, \tag{3.6}$$

where vectors $\mathbf{g^i}$ form a contravariant basis, aligned to the grid lines (ξ, η, τ) in the physical domain:

$$\mathbf{g^1} = (\frac{\partial x}{\partial \xi}, \frac{\partial y}{\partial \xi}, \frac{\partial z}{\partial \xi}),\ \mathbf{g^2} = (\frac{\partial x}{\partial \eta}, \frac{\partial y}{\partial \eta}, \frac{\partial z}{\partial \eta}),\ \mathbf{g^3} = (\frac{\partial x}{\partial \tau}, \frac{\partial y}{\partial \tau}, \frac{\partial z}{\partial \tau}). \tag{3.7}$$

Since we are interested to represent $\vec{u}$ in the computational domain Ω' with canonical cells (*i.e.*, orthonormal cells with spacings equal to one), the arbitrary unit basis is defined as

$$\mathbf{g}^{(\mathbf{1})} = \frac{1}{h_1}\mathbf{g}^{\mathbf{1}}, \quad \mathbf{g}^{(\mathbf{2})} = \frac{1}{h_2}\mathbf{g}^{\mathbf{2}}, \quad \mathbf{g}^{(\mathbf{3})} = \frac{1}{h_3}\mathbf{g}^{\mathbf{3}}, \tag{3.8}$$

where the h_i terms are the scale factors, given by the lengths of $\{\mathbf{g}^{\mathbf{i}}\}$:

$$\mathbf{h_i} = ((\mathbf{g}^{\mathbf{i}})^T \cdot \mathbf{g}^{\mathbf{i}})^{\frac{1}{2}}. \tag{3.9}$$

Thus, a transformation that maps a vector defined in an arbitrary unit basis to the canonical Cartesian basis $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ is defined by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \mathbf{g}_{\mathbf{x}}^{(\mathbf{1})} & \mathbf{g}_{\mathbf{x}}^{(\mathbf{2})} & \mathbf{g}_{\mathbf{x}}^{(\mathbf{3})} \\ \mathbf{g}_{\mathbf{y}}^{(\mathbf{1})} & \mathbf{g}_{\mathbf{y}}^{(\mathbf{2})} & \mathbf{g}_{\mathbf{y}}^{(\mathbf{3})} \\ \mathbf{g}_{\mathbf{z}}^{(\mathbf{1})} & \mathbf{g}_{\mathbf{z}}^{(\mathbf{2})} & \mathbf{g}_{\mathbf{z}}^{(\mathbf{3})} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, \tag{3.10}$$

and it is valid only if the 3x3 matrix (T_{cp}) shown in Eq. (3.10) is non-singular, *i.e.*, the vectors $\{\mathbf{g}^{\mathbf{i}}\}$ are linearly independent (which is always the case for grids of interest). Converting from the Cartesian to an arbitrary unit basis is obtaining by inverting Eq. (3.10):

$$\vec{u}_{\xi\eta\tau} = T_{cp}^{-1}\vec{u}_{ijk}. \tag{3.11}$$

is used.

We illustrate the transformation process for two dimensions in Figure 3.2. First, the vector $\vec{u}_{ij}$ on the physical domain (Figure 3.2a) is expressed in the Cartesian basis (Figure 3.2b). Then, $\vec{u}_{ij}$ is decomposed in an arbitrary basis (Figure 3.2c) by applying Eq. (3.11). Each new component (u_1 and u_2) can be seen as the projection of $\vec{u}_{ij}$ onto the vectors tangent to the grid lines ($\mathbf{g}^{\mathbf{2}}$ and $\mathbf{g}^{\mathbf{1}}$, respectively). The resulting vector $\vec{u}_{\xi\eta}$ is the representation of $\vec{u}_{ij}$ in the computational domain (Figure 3.2d).

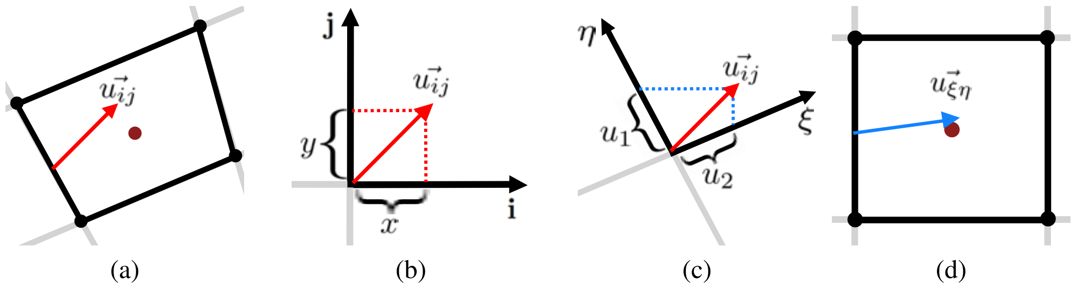


Figure 3.2: Transformation of vector $\vec{u}_{ij}$ to $\vec{u}_{\xi\eta}$. (a) $\vec{u}_{ij}$ represented on the physical curvilinear cell; (b) $\vec{u}_{ij}$ represented in Cartesian coordinates; (c) projections of $\vec{u}_i j$ onto the basis of the curvilinear cell; (d) $\vec{u}_{\xi\eta}$ represented in the computational domain.

To evaluate the Semi-Lagrangian algorithm in the computational grid, we only need to transform the velocity from the physical domain to the computational domain (the mathematical proof is given in Appendix A.3). In the computational domain, we use Eq. (3.4) with $\vec{v}^n = \vec{u}^n_{\xi\eta}$ to obtain the position $x_{\xi\eta\tau}$. Then, the intermediary velocity field estimated at $x_{\xi\eta\tau}$ is used as the new intermediate velocity field $\vec{u}_{ij}^{*(n+1)}$ (Eq. (3.3)). Notice that the transformed velocity is **only used in the back-trajectory step**; the interpolation at $x_{\xi\eta\tau}$ uses velocities defined in Cartesian coordinates. This allows us to directly adapt higher-order methods, such as the modified MacCormack (SELLE et al., 2008). The complete curvilinear-grid semi-Lagrangian method is summarized in Algorithm 1.

Algorithm 1 Curvilinear-grid semi-Lagrangian method

```
/* Cartesian velocities (ijk basis) stored at the centers of the faces of the physical grid, in a staggered way */
for all grid cells in the physical domain do
    /* At the center of each cell face, obtain the full velocity vector by interpolating the other components */
    $\tilde{u}_{ijk} \leftarrow interpolateVelocityField(\vec{u}^n_{ijk}, x)$
    /* Transform the full velocity vector to the $\xi\eta\tau$ basis */
    $\tilde{u}_{\xi\eta\tau} \leftarrow transformToComputationalGrid(\tilde{u}_{ijk})$
    /* Compute $x_p$, in the previous time step */
    $x_{\xi\eta\tau} \leftarrow findPreviousPosition\ (x, \tilde{u}_{\xi\eta\tau}, \Delta t)$
    /* Estimate the intermediate velocity field $u^{*(n+1)}_{ijk}$ */
    $u^{*(n+1)}_{ijk}(x) \leftarrow interpolateVelocityField(\vec{u}^n_{ijk}, x_{\xi\eta\tau})$
end for
```

3.2 Pressure Solving on Curvilinear Grids

In computer graphics, the standard way of representing *mass conservation* for incompressible fluids is by enforcing that the divergence of the velocity field is zero (Eq. (3.2)). However, the integral form:

$$\int_V \nabla \cdot \vec{u}\ \mathrm{d}V = \int_S \vec{u} \cdot \vec{n}\ \mathrm{d}S = 0, \tag{3.12}$$

is more convenient for our solution, as it simplifies the representation of divergence and gradient operators on general coordinate systems. Moreover, if we choose to solve all the equations using the differential representation (i.e. mass conservation as $\nabla \cdot \vec{u} = 0$), additional non-conservative terms, called Christoffel Symbols, are needed to account for the non-straight space axis deformations (KOSHIZUKA; OKA; KONDO, 1990).

In Eq. (3.12), V and S denote the volume and the faces of the cell under evaluation, respectively. Using the projection method to couple momentum (Eq. (3.1)) and mass conservation (Eq. (3.12)) equations yields

$$\int_S \vec{u}^{(n+1)} \cdot \vec{n}\ \mathrm{d}S = \int_S \vec{u}^{*(n+1)} \cdot \vec{n}\ \mathrm{d}S - \Delta t \int_S \frac{\partial p^{(n+1)}}{\partial \vec{n}}\ \mathrm{d}S = 0, \tag{3.13}$$

where $\vec{n}$ represents the faces' corresponding unit normal vectors.

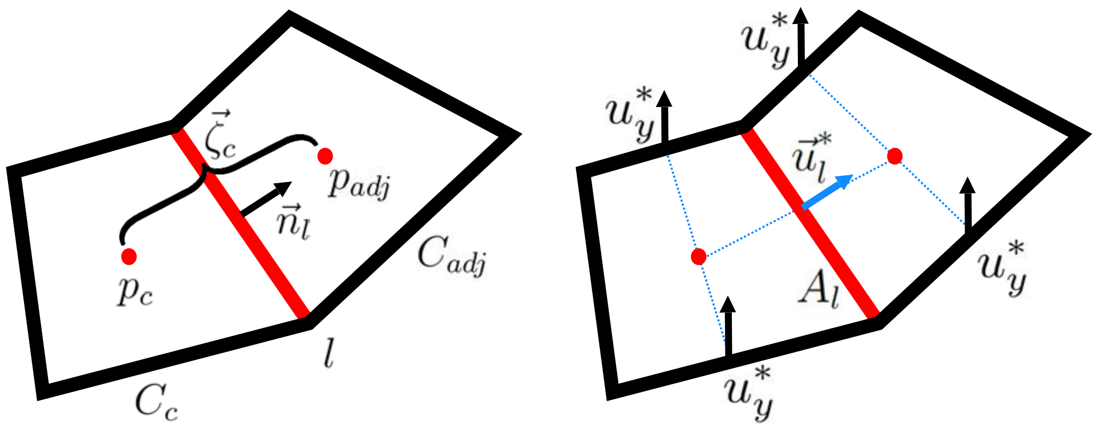
The pressure field that guarantees mass conservation is obtained by solving

$$\int_S \frac{\partial p^{(n+1)}}{\partial \vec{n}}\ \mathrm{d}S = \frac{\int_S \vec{u}^{*(n+1)} \cdot \vec{n}\ \mathrm{d}S}{\Delta t}. \tag{3.14}$$

The pressure derivatives on the left-hand side of Eq. (3.14) are discretized as

$$\int_S \frac{\partial p^{(n+1)}}{\partial \vec{n}}\ \mathrm{d}S \approx \left[(p^{(n+1)}_c - p^{(n+1)}_{adj}) \frac{(\vec{\zeta}_c \cdot \vec{n}_l)}{||\vec{\zeta}_c||} \right] A_l, \tag{3.15}$$

where p_c is the pressure evaluated in the current cell C_c, and p_{adj} is the pressure evaluated in the adjacent cell C_{adj} that shares face l with C_c. A_l is l's area, and $\vec{\zeta}_c$ is the vector



(a) Pressure derivative along cell face normal (b) Velocity interpolation

Figure 3.3: (a) The pressure derivative along a cell face normal and the elements that compose it. (b) Velocity interpolation on the center of a cell face. Due the staggered velocity scheme, the vertical velocity is interpolated from nearby locations (u_y^*) to form the interpolated velocity $\vec{u}_l^* = (u_x, \tilde{u}_y)$.

which connects the centers of cells C_c and C_{adj}. Figure 3.3a show the representation of the pressure derivative along the normal of a cell face.

In this discretization, we dropped the cross-derivative terms that affect each flux of the cell faces. Fully evaluating the original system would yield a 9-point Laplace-matrix discretization for 2-D and a 27-point Laplace-matrix for 3-D. The same approach was adopted in (SHYY; VU, 1991), where the authors point out that dropping these terms does not affect the accuracy of the solution. A detailed derivation of the pressure system is given by Appendix A.4.

Finally, we discretize the cell fluxes on the right-hand side of Eq. (3.14) as

$$\int_S \vec{u}^{*(n+1)} \cdot \vec{n} \ \mathrm{d}S \approx \sum_{l \,\in\, \text{faces}} (\vec{u}_l^{*(n+1)} \cdot \vec{n}_l) A_l, \tag{3.16}$$

where the intermediate velocity $\vec{u}_l^{*(n+1)}$ is evaluated at the center of the cell face l. The components of $\vec{u}_l^{*(n+1)}$ that are not known a-priori are interpolated from the cell's neighbors in the same fashion as in regular grids. The interpolation process is illustrated in Figure 3.3b.

3.3 Velocity Projection on Curvilinear Grids

When substituting in Eq. (3.13) the pressure field obtained with the solution of Eq. (3.14), one obtains the fluxes (across the cells' faces) that satisfy the incompressibility constraint. Since we only store the Cartesian components of the velocity in the center of each cell face, we need to transform back the projected fluxes to velocity components. Shyy and Vu (SHYY; VU, 1991) recover the velocity components using an iterative scheme based on D'Yakunov's method (CONCUS; GOLUB, 1972). Although accurate, such method tends to be slow, making it less attractive for use with grids containing a large number of cells.

We introduce a novel and efficient approach for updating (projecting) the Cartesian components of the velocity field on staggered curvilinear grids. Our solution is faster, simpler, and easier to implement than D'Yakunov's method. *It consists of correcting the unprojected velocity field directly by evaluating pressure derivatives in all Cartesian directions*. Using the chain rule, we can write:

$$
\begin{aligned}
\vec{u}_x &= \vec{u}^*_x - \Delta t \frac{\partial p}{\partial x} = \vec{u}^*_x - \Delta t(\frac{\partial p}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \tilde{p}}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial \tilde{p}}{\partial \tau}\frac{\partial \tau}{\partial x})\mathbf{i}, \\
\vec{u}_y &= \vec{u}^*_y - \Delta t \frac{\partial p}{\partial y} = \vec{u}^*_y - \Delta t(\frac{\partial p}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial \tilde{p}}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial \tilde{p}}{\partial \tau}\frac{\partial \tau}{\partial y})\mathbf{j}, \\
\vec{u}_z &= \vec{u}^*_z - \Delta t \frac{\partial p}{\partial z} = \vec{u}^*_z - \Delta t(\frac{\partial p}{\partial \tau}\frac{\partial \tau}{\partial z} + \frac{\partial \tilde{p}}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial \tilde{p}}{\partial \eta}\frac{\partial \eta}{\partial z})\mathbf{k},
\end{aligned}
\tag{3.17}
$$

where (ξ, η, τ) are the grid lines in the physical domain (Section 3.1). The 2-D version of these equations just drop the third term (in τ) inside the parenthesis.

Recall that in the Cartesian-staggered arrangement, each cell face l stores a single Cartesian velocity component, and Eq. (3.17) is used to update all of them. Here, we describe the procedure for updating the velocity component along the x direction (highlighted arrows in Figs. 3.4a and 3.4b); updating other components is similar.

For a given cell face l, the term $\frac{\partial p}{\partial \xi}$ is computed directly from the pressure values stored at the centers of the two cells sharing l (red dots in the Figure 3.4a). Computing the term $\frac{\partial p}{\partial \eta}$, however, requires pressure values at the vertices (end points) of l, which are not readily available (solid blue dots in Figure 3.4b). Each such value is interpolated from the pressure values stored at the centers of all cells sharing the corresponding vertex (shown as small blue circles in Figure 3.4b). Intuitively, we compensate the lack "de-interpolation" of the Cartesian velocity components traditionally used in CFD with a few additional interpolations on the pressure field.

Our approach has some desirable properties: (i) for regular grids, it reverts back to the standard MAC formulation; (ii) for any orthogonal grid, $\frac{\partial x}{\partial \xi} + \frac{\partial y}{\partial \eta} = 1$. The first property allows our method to be used on regular grids without any adaptation. The second, shows that the contribution of the pressure correction will be physically consistent to orthogonal grids. All simulations (both 2-D and 3-D) shown in the thesis and accompanying video were produced using our projection technique.

3.4 Adaptive Path Integrator

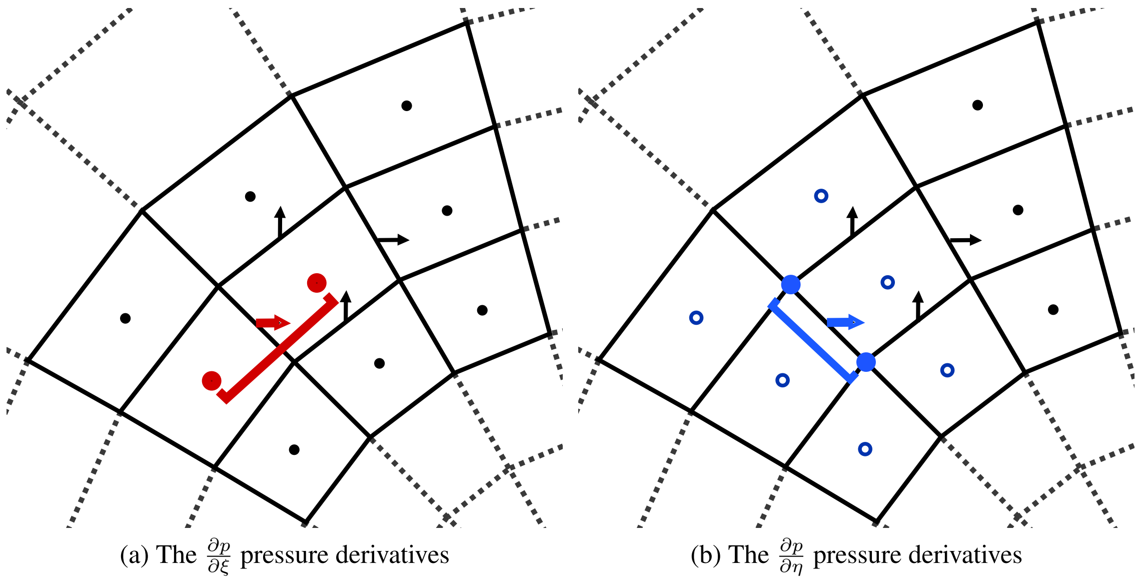
The proposed non-regular semi-Lagrangian method has a CFL restriction. Since it estimates the particle trajectory by evaluating local metrics, it can incorrectly estimate the backtracked particle path if the CFL condition is bigger than 1 (BARROSO; CELES, 2011). To alleviate this condition, we adopt a multi-step adaptive Runge-Kutta integrator, which enforces that the local CFL condition for the advection step is always less than 1. Thus, the adaptive path integrator time-step δt is defined as a function of the global time-step Δt as

$$\delta t = min(\kappa, \Delta t), \tag{3.18}$$

with

$$\kappa = min(\frac{h_1}{|\vec{u}_x|}, \frac{h_2}{|\vec{u}_y|}, \frac{h_3}{|\vec{u}_z|}) \tag{3.19}$$

where h_1, h_2 and h_3 are the local cell scale factors defined in Eq. (3.9).



(a) The $\frac{\partial p}{\partial \xi}$ pressure derivatives (b) The $\frac{\partial p}{\partial \eta}$ pressure derivatives

Figure 3.4: Updating the x-component of the Cartesian velocity (highlighted arrow) at a given cell face l. (a) Pressure derivatives evaluated from values stored at the centers of the two cells sharing l (red dots). (b) Pressure derivatives evaluated from interpolated values (solid blue dots) computed from the pressure values stored at the centers of the surrounding cells (blue circles).

Then we integrate the trajectory of the particle using δt until it reaches the same amount of time expressed by Δt, using the Runge-Kutta half-step method (Eq. (A.33)). Since curvilinear grids have cells with different sizes, the number of steps required for different cells may vary along the grid.

3.5 Domain Decomposition with Overlapping Grids

In order to efficiently support multiple objects with independent and dynamic rigid-body motions, we decompose the simulation domain using *overlapping grids* (BRANDT, 1977; CHESSHIRE; HENSHAW, 1990). Each object is represented by a curvilinear grid that tightly fits its boundary and condensates cells in regions where higher vorticity is expected. Such grids are then superimposed on a regular one that delimits the simulation domain. A conceptual representation of out overlapping grids setup is shown in Figure 3.5. This approach lends to good refinement, allowing for optimal use of computational resources.

Dobashi et al. (DOBASHI et al., 2008) use overlapping grids to produce flow animations. Thus, conceptually, our ovelapping-grid solution is similar to theirs. However, they use regular grids both for the background and for the superimposed obstacles. As such, boundary conditions cannot be properly represented and the resulting simulations share the same limitations as the conventional regular-grid ones.

We start evaluating the advection phase on the background grid, and interpolate the intermediary velocity values (*i.e.*, $\vec{u}^{*(n+1)}$ in Eq. (3.3)) at the boundaries of the curvilinear grids. The advection phase then proceeds inside the curvilinear grids, and the resulting intermediary velocity field is interpolated back to the background grid. The new pressure field is obtained using a multigrid solver (Chapter 4). After the pressure-solving step, each grid is then projected independently.

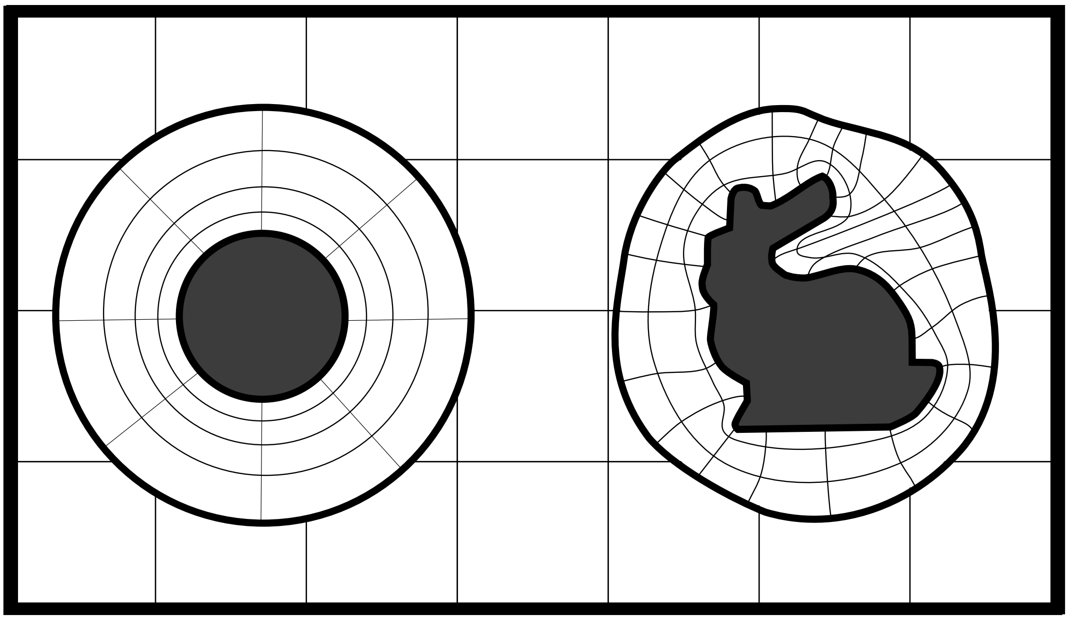


Figure 3.5: Conceptual representation of the overlapping grids configuration. A regular grid discretizes the simulation domain and each object has an attached body-fitted grid.

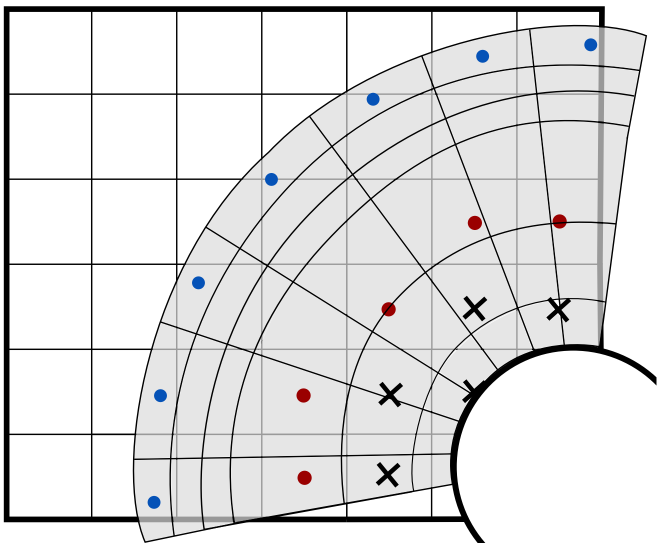


Figure 3.6: Domain decomposition using overlapping grids. An underlying regular grid delimits the simulation domain. A curvilinear circular grid (light gray) overlaps it. Interpolation cells are used for grid communication and for setting boundary conditions. Blue dots indicate curvilinear-grid interpolation cells; red dots denote the regular-grid interpolation cells; unused cells are marked with $\times$.

The use of overlapping grids requires the simulation to transfer flow information between the different grids . For this, we set a band of cells in which both grids slightly overlap. This guarantees that the interpolation points evaluate fluid properties only where they are correctly solved. We classify each cell of both regular and curvilinear grids as either *discretization*, *interpolation* or *unused* (Figure 3.6). *Interpolation cells* are the ones at the edges of the overlapping bands. Their purpose is to sample the values stored in cells of the grid they overlap with, providing the actual mechanism for exchanging information between grids. We adopted Dirichlet boundary conditions, which are based on explicit specification of boundary values. Given an interpolation band, its outer edge consists of curvilinear-grid interpolation cells, while the inner edge is made of regular-grid interpolation cells. Figure 3.6 indicates curvilinear-grid interpolation cells with blue dots, and regular-grid interpolation cells with red dots. *Unused cells* are regular-grid cells that overlap with the objects' grids, except for the ones in the overlapping band, and are ignored by the flow solver. They are indicated with an $\times$ in Figure 3.6. All remaining cells are discretization ones. These, along with the interpolation cells, are on which the simulation is performed.

The optimal size of the overlapping band depends on the used interpolation scheme, and on the sizes of the grid cells. Although more sophisticate interpolation schemes can be used, according to our experience a band with 3 to 4 cells produces good results with linear interpolation. Better results are obtained by refining the curvilinear cells in the overlapping band so that the area of such a cell is at most half of the area of the underlying regular cell.

In this Chapter we presented the details associated with the three steps of our projection method for curvilinear grids. In Section 3.1, an efficient algorithm for advection for curvilinear grids was described. In Section 3.2, we presented the formulation used in the pressure system step. In Section 3.3, we presented the details of our projection method, which supports Cartersian staggered variable arrangement. Finally, in Section 3.5 we described the domain decomposition technique, which facilitates the grid generation and increases the applicability of our method.

4 MULTIGRID ON CURVILINEAR GRIDS AND GRID GENERATION

In the following subsections we will give a detailed description of the Multigrid algorithm, grid generation techniques, cell mapping and grid cutting, adaptive path integrator algorithm.

The *multigrid method* is a fast algorithm for solving linear systems (BRIGGS; HENSON; MCCORMICK, 2000). Due its capacity to solve linear system with estimated $O(n)$ computational complexity, it is widely employed in computer graphics applications (COHEN; TARIQ; GREEN, 2010; MCADAMS; SIFAKIS; TERAN, 2010; CHENTANEZ; MULLER, 2011b).

For regular grids, the matrix coefficients (operators) of the linear system are copied from the finest to the coarsest levels. On curvilinear grids, however, the coarsening process degenerates the shape of the discretized object (Figure 4.1). Thus, simply copying the values of the matrix coefficients affects the method's convergence. To improve it, we use the scheme described in (HE et al., 1996). In such a scheme, the areas and volumes of the coarse-grid cells are defined as the sum of the corresponding cells in the finest grid, and the coarse operators are defined by the area-mean of each operator in the finest grid. Thus, the coarse grid areas A_c and volumes V_c are defined by:

$$\begin{aligned} A_c &= \sum_{i \,\in\, finest} A_f^i, \\ V_c &= \sum_{i \,\in\, finest} V_f^i, \end{aligned} \tag{4.1}$$

where the A_f^i and V_f^i are the correspondent fine grid areas and volumes, respectively. The coarse grid operator C_c is the area mean of each fine cell operator C_f^i:

$$C_c = \frac{1}{A_c} \sum_{i \,\in\, finest} C_f^i A_f^i, \tag{4.2}$$

since the Poisson matrix is defined as function of the cell areas (Eq. (3.15)).

Our multigrid implementation uses the *incomplete multigrid method* (ICMG) (HINATSU; FERZIGER, 1991) with Dirichlet boundary conditions. ICMG constrains the communication between different domains to the finest level; it contrasts with the *complete multigrid method* (CCMG) which exchanges information among all grid levels. CCMG is harder to implement, due the nature of the coarsening operator that may generate complex interpolation configurations which require special treatment (HENSHAW, 2005b).

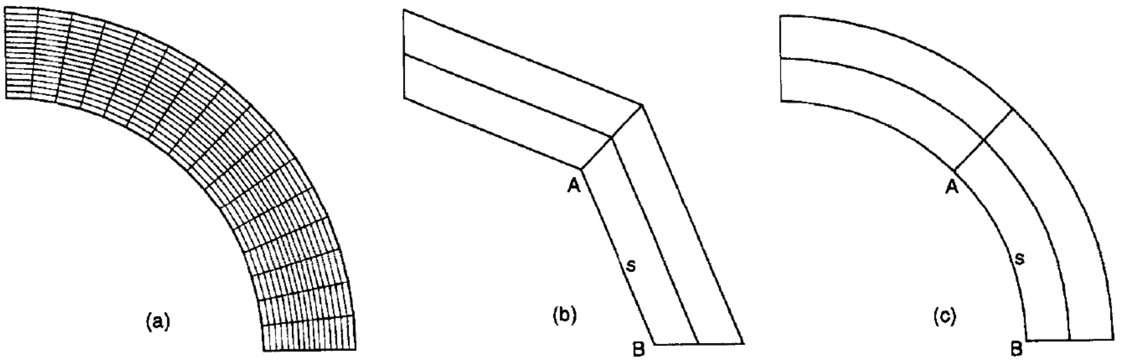


Figure 4.1: Coarsening of a curvilinear grid, extracted from (HE et al., 1996). (a) Finest circular grid. (b) Coarse circular grid. (c) Coarse grid corrected with fine grid information.

Exchanging interpolated pressure information between overlapping domains may introduce discontinuities in the resulting pressure fields, most notably when the interpolation direction is going from the foreground to the background grid. Thus, at each iteration of the pressure solver, we use Henshaw's approach (HENSHAW, 2005b), which consists of applying additional Gauss-Seidel iterations on neighborhoods of the background boundary cells. This process smooths out the high-frequency error modes that may appear due the interpolation process.

Since different grids on the simulation domain may converge at different rates, we dynamically adjust the number of smoothing iterations (ν) for each grid based on its residual error. We compare the residual ratios between the foreground and background grids res_{bg}/res_{fg} to $\sigma^{1/\nu}$, since it allows larger values of ν to change more easily. The residual ratio σ is defined within lower (σ_-) and upper (σ_+) boundaries. Numerical experimentation of (HENSHAW, 2005b) defined $(\sigma_-, \sigma_+) = (\frac{1}{2}, 2)$.

Algorithm 2 summarizes all the steps performed on our multigrid algorithm.

4.1 Cell Mapping and Grid Cutting

Given a position expressed in world coordinates, finding the cell that contains it on the regular background grid is a straightforward process. This can be expressed by

$$[ijk] = \frac{v_{pos} - v_{bg}}{dx_{bg}}, \tag{4.3}$$

where i, j, k are the background cells indexes, v_{pos} is the position vector in world coordinates, v_{bg} is the origin of the background grid expressed in the world coordinate system, and dx_{bg} is the background grid spacing. This formula only holds for regularly-spaced grids with constant curvature. The process of finding the cell on curvilinear grids that contains a point in world coordinates can become complex.

In our solution, interpolation is only needed for boundary cells of the background grid (Section 3.5). Therefore, we mapped background boundary cells to their closest respective foreground grid cells. In this context, "closest" is defined as the distance between the foreground and background grid cell centers. Since it is more efficient to find a correspondent cell in the background grid, we iterate through a band of outer foreground grid cells, looking for their correspondent in the background grid, using Eq. (4.3). We summarize the cell mapping process in the Algorithm 3.

Algorithm 2 Multigrid on overlapping domains

```
while $res_{bg} + res_{fg} < res_{max}$ do
    perform a full V-Cycle on the background grid
    $res_{bg} \leftarrow$ calculateResidual(backgroundGrid)
    /* Interpolate pressures from the background to the foreground overlapped grid */
    for all cells on the boundary of the foreground grid do
        $p_{fg} \leftarrow interpolatePressureField(p_{bg}, x_{fg})$
    end for
    perform $\nu$ full V-Cycles on the foreground grid
    $res_{fg} \leftarrow$ calculateResidual(foregroundGrid)
    /* Interpolate pressures from the foreground overlapped to the background grid */
    for all boundary cells on the background grid do
        $p_{bg} \leftarrow interpolatePressureField(p_{fg}, x_{bg})$
    end for
    /* Remove high frequencies that may appear due interpolation.*/
    perform smoothBoundaryCells(backgroundGrid, numSmooths)
    /* Adjust the number of sub-smooths on the foreground grid*/
    if $res_{bg}/res_{fg} < \sigma_{-}^{1/\nu}$ do
        $\nu \leftarrow min(1, \nu - 1)$            /* Decrease number of sub-smooths */
    else if $res_{bg}/res_{fg} > \sigma_{+}^{1/\nu}$ do
        $\nu \leftarrow max(\nu + 1, num_{max})$ /* Increase number of sub-smooths */
    end if
end while
```

Algorithm 3 Foreground grid cells mapping

```
/* Verify if the foreground grid has moved relative to background grid */
if $hasChanged(pos_{fg} - pos_{bg})$ do
    for nOuterCells within an outer band of the foreground grid do
        /* Find by Eq. (4.3) the background grid index correpondent to the foreground grid cell center*/
        $[i_{bg}\ j_{bg}\ k_{bg}] \leftarrow findBgGridIndex(cellCenter_{fg})$
        /* If the distances between the cells centers are the smallest encountered so far*/
        if $cellCenter_{fg} - cellCenter_{bg} < distancesMap(i_{bg}, j_{bg}, k_{bg})$ do
            $distancesMap(i_{bg}, j_{bg}, k_{bg}) \leftarrow cellCenter_{fg} - cellCenter_{bg}$
            /* Map the foreground grid cell to the background grid cell*/
            $cellsMap(i_{bg}, j_{bg}, k_{bg}) \leftarrow index_{fg}$
        end if
    end for
end if
```

The *nOuterCells* parameter in Algorithm 3 is defined empirically, and vary with background and foreground grid configurations. It depends on the foreground grid shape and the aspect ratio between foreground and background grid cells.

The cells from the background grid which remained inside the boundary overlapped interior grid cells must be removed (black cells on Figure 3.6). Considering that the boundary cells are always connected - a boundary cell will always have at least a neighbor that is also a boundary cell - we use simple line filling algorithm to remove unused cells. We notice that there are better strategies that can be implemented to remove the background unused cells. For example, one could use OpenGL to rasterize the grids and identify unused cells. A similar method was employed in (CRANE; LLAMAS; TARIQ, 2007).

4.2 Grid Generation Techniques

Since our method supports overlapped grids, the process of grid generation is greatly simplified (WYMAN, 2001). In this section, we will present the steps used to generate a 2-D grid corresponding to a profile of the Stanford bunny, using the extrusion of the normal vectors. For this example, we use Gridgen by Pointwise Inc. (POINTWISE, 2012), but other software grid generation software could have been used instead.

Since Gridgen works with parametric lines, we had to export the bunny geometry to a NURBS representation. We used 3D Studio Max for object modeling and exported the geometry to IGES format (IGES, 2006). All imported objects in Gridgen are initially categorized as database entities. The user must convert these entities into connectors, upon which grid points can be defined. The number of points present on the connector (Figure 4.2) define the number of cells along one dimension of the grid. This enables concentration of points in regions of higher turbulence or vorticity.

To generate the grid with Gridgen, the extrude normals command is used (Figure 4.3). The user can configure parameters to manipulate the extrusion process: the number of steps define the amount of cells which will follow the normal extrusion path; initial Δs is the initial cell size along the normal direction; the growth rate defines the scaling parameter that will be incrementally applied to the initial size and the smoothing parameters define the relaxation configuration which prevents self-overlapping. The cell extrusion configuration can generate cells which overlap (Figure 4.4), depending on the set of parameters. Thus, if an invalid grid is generated, the user must provide a different configuration, and re-run the extrude normals command (Figure 4.5).

We summarize the creation of grids based on the extrusion of object normals using Gridgen in the following steps:

- Import the parametric line/surface object representation into Gridgen (input is a NURBS representation);
- Select the object and convert it from database entities to connectors;
- Select the object connector and specify the number of points in the base dimension;
- Select the object connector and use the Create→Extrude→Normal command.
- Configure number of steps, initial Δs, growth rate and smoothing parameters that will be used in the extrusion process.

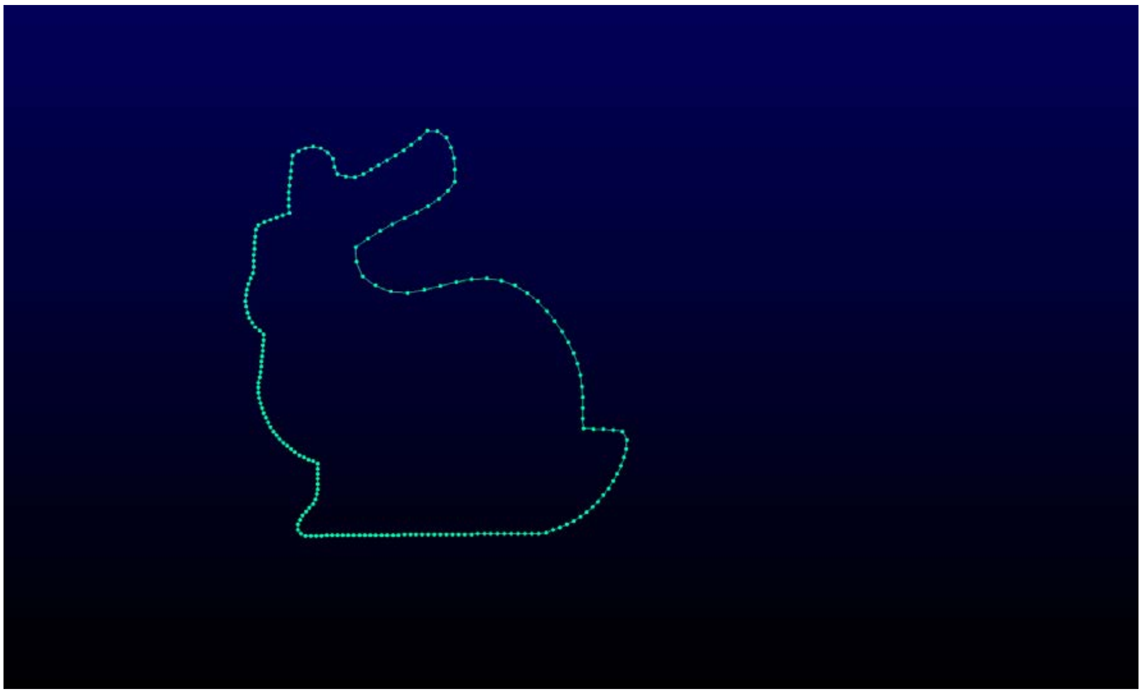
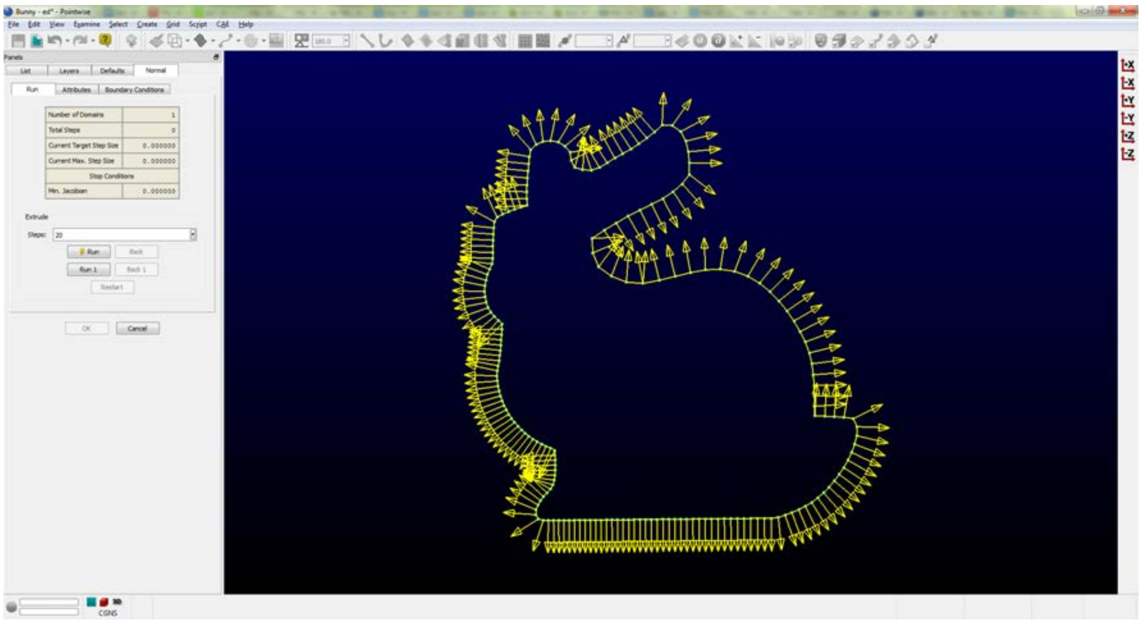


Figure 4.2: Profile of the Stanford bunny defined by a series of points, using the software Gridgen, by Pointwise, Inc. A 2-D mesh will be obtained by extruding each point along the direction of its corresponding normal vector. The user can condensate more points (and in turn, cells) in regions of expected turbulence.


Bunny - ed* - Pointwise
File Edit View Examine Select Create Grid Script CAE Help
Panels
List Layers Defaults Normal
Run Attributes Boundary Conditions
Run Back
Run 1 Back 1
Restart
OK Cancel
CGNS

Number of Domains	1
Total Steps	0
Current Target Step Size	0.000000
Current Max. Step Size	0.000000
Stop Conditions	
Min. Jacobian	0.000000

Extrude

Steps: 20

Figure 4.3: Bunny normals: grid lines will be generated along the normals.

- Extrude normals. If some overlapping occurs, reconfigure the parameters of the last step and re-run the extrusion command.

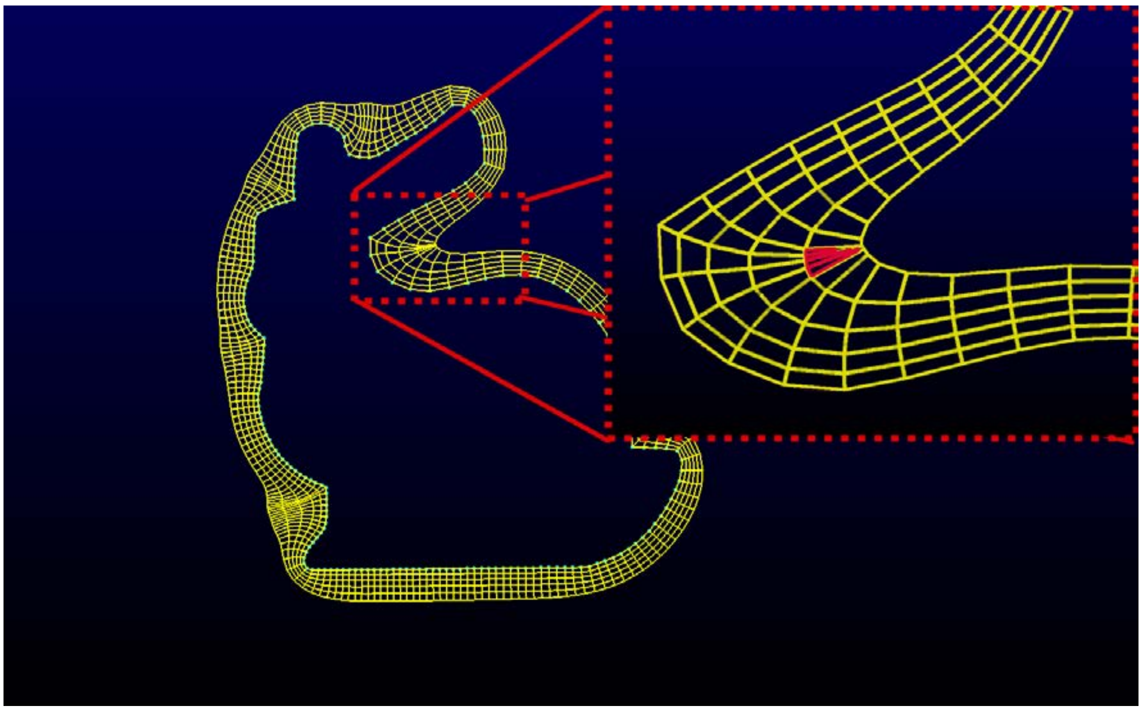


Figure 4.4: Example of a bad grid (containing overlapping cells) resulting from the specification of improper parameter values. Parameters used: number of steps: 20; Δs : 0.008; growth rate: 1.02; smoothing parameter: 0.5.

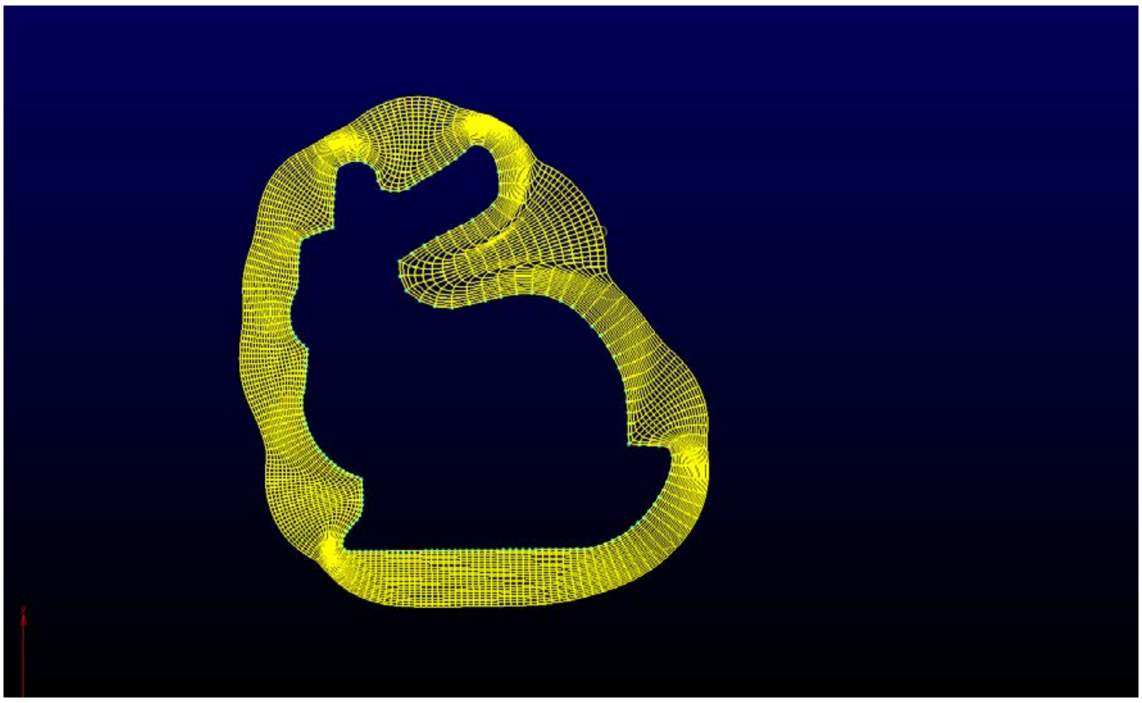


Figure 4.5: Bunny grid: 2-D Bunny grid obtained with the specification of correct parameter values. Parameters used: number of steps: 20; Δs : 0.008; growth rate: 1.02; smoothing parameter: 0.9.

5 RESULTS

We have implemented the techniques described in this thesis using C++. For the pressure-solving step, we have both a GPU and a CPU implementations. The GPU implementation, based on CUDA, uses the conjugate gradient solver of the CUSP libraries (BELL; GARLAND, 2012). The CPU version uses multigrid and currently is available only for 2-D. As support libraries, we adopted Thrust (BELL; HOBEROCK, 2012) for facilitating the implementation of parallel algorithms, NVIDIA's PhysX SDK (NVIDIA, 2012a) for simulating rigid body dynamics, Cg Toolkit (NVIDIA, 2012b) for loading and manipulating shaders, and the physically based ray tracer (PBRT) (PHARR; HUMPHREYS, 2012) for visualizing the results.

To maintain a cohesive workspace for the code, a namespace called "Chimera" was created. Within this namespace, three subprojects were developed: *Chimera Core* - responsible for the low-level configuration of libraries and definitions; *Chimera Math* - containing the mathematical framework and support functions that are needed for solving the underlying equations of fluid flows; and *Chimera Foundation* - responsible several auxiliary functions which are responsible for grid loading, rendering utilities, resource management, etc. Since the nature of interaction is inherently different between two and three dimensions, two distinct projects were created: *Chimera 2D* and *Chimera 3D*.

We render our scenes using PBRT combined with the Luxrender frontend (LUXRENDER, 2012). Our animation pipeline is the following: first, we setup the scene in 3Ds Max or Blender (Figure 5.1). The grids are generated, exported and loaded into our application. Then, the scene is setup for simulation: boundary conditions, smoke sources and solver parameters are set (Figure 5.2). The simulation is performed and its output is saved. Finally, the output is loaded into Luxrender (Figure 5.3) and the rendering process continues.

5.1 Results

This section presents the results obtained with our technique, comparing it with traditional regular grids. We did not compare the results of our approach with the ones obtained with standard variational boundary condition method (BATTY; BERTAILS; BRIDSON, 2007), because this method is not able to maintain the correct aerodynamic properties explored in our scenes. Although unstructured grids (KLINGNER et al., 2006) can also correctly represent boundary conditions, they are inherently much slower than our approach and, as such, have not been included in this comparison.

Figures 5.4 and 5.5 demonstrate the ability of our approach to represent proper boundary conditions. They compare our simulation results against the ones obtained using a regular grid for a cross section of an airfoil. This is a quite sensitive example due to

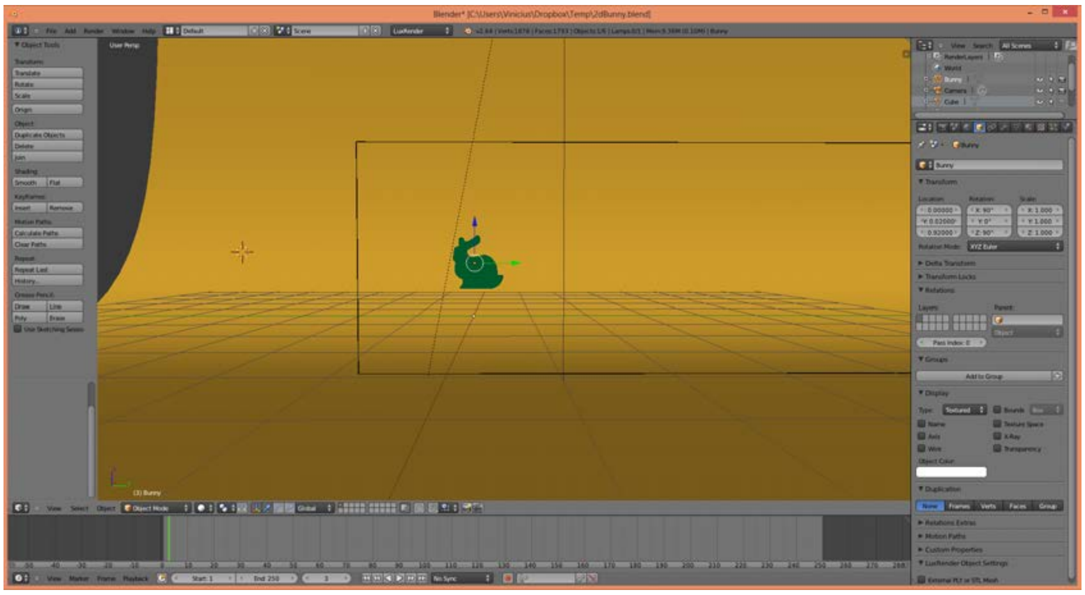
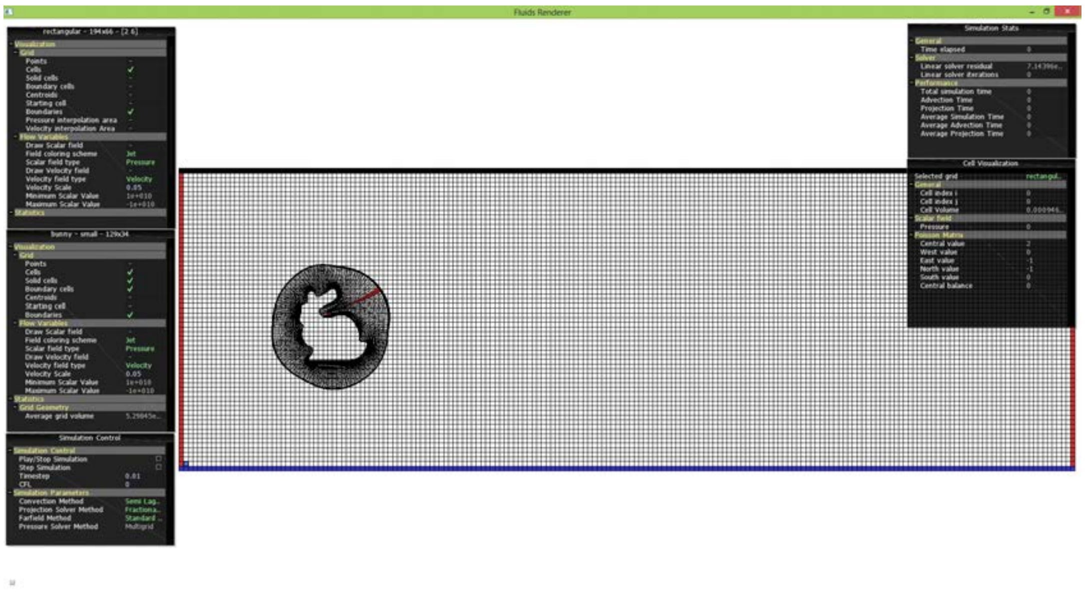


Figure 5.1: Animation pipeline: setup of the scene on Blender.



```
rectangular - 194x66 - [2 6]
- Visualization
 - Grid
    Points                          -
    Cells                           ✓
    Solid cells                     -
    Boundary cells                  -
    Centroids                       -
    Starting cell                   -
    Boundaries                      ✓
    Pressure interpolation area     -
    Velocity interpolation Area     -
 - Flow Variables
    Draw Scalar field               -
    Field coloring scheme           Jet
    Scalar field type               Pressure
    Draw Velocity field             -
    Velocity field type             Velocity
    Velocity Scale                  0.05
    Minimum Scalar Value            1e+010
    Maximum Scalar Value            -1e+010
- Statistics

bunny - small - 129x34
- Visualization
 - Grid
    Points                          -
    Cells                           ✓
    Solid cells                     ✓
    Boundary cells                  ✓
    Centroids                       -
    Starting cell                   -
    Boundaries                      ✓
 - Flow Variables
    Draw Scalar field               -
    Field coloring scheme           Jet
    Scalar field type               Pressure
    Draw Velocity field             -
    Velocity field type             Velocity
    Velocity Scale                  0.05
    Minimum Scalar Value            1e+010
    Maximum Scalar Value            -1e+010
- Statistics
 - Grid Geometry
    Average grid volume             5.29845e...

Simulation Control
- Simulation Control
   Play/Stop Simulation             ☐
   Step Simulation                  ☐
   Timestep                         0.01
   CFL                              0
- Simulation Parameters
   Convection Method                Semi Lag...
   Projection Solver Method         Fractiona...
   Farfield Method                  Standard ...
   Pressure Solver Method           Multigrid

Simulation Stats
- General
   Time elapsed                     0
- Solver
   Linear solver residual           7.14396e...
   Linear solver iterations         0
- Performance
   Total simulation time            0
   Advection Time                   0
   Projection Time                  0
   Average Simulation Time          0
   Average Advection Time           0
   Average Projection Time          0

Cell Visualization
 Selected grid                      rectangul...
- General
   Cell index i                     0
   Cell index j                     0
   Cell Volume                      0.000946...
- Scalar field
   Pressure                         0
- Poisson Matrix
   Central value                    2
   West value                       0
   East value                       -1
   North value                      -1
   South value                      0
   Central balance                  0
```

Figure 5.2: Animation pipeline: setup of the scene on our simulation application.

obstacle's fine aerodynamics properties. Figure 5.4 shows the simulation results for a horizontally-oriented airfoil at a given moment. The image on the left was obtained using a regular grid. Note how the poorly-enforced boundary conditions resulting from the object's discretization incorrectly generates high-vorticity regions behind the airfoil. In contrast, the results of our simulation, seen on the right, show a well-behaved stream of smoke.

Figure 5.5 shows a similar comparison with the airfoil oriented at 10 degrees. Again, the discretization of the object's boundary due to the use of a regular grid leads to a completely incorrect simulation (left). Our results can be seen on the right. The accompanying video shows that in our simulation the air flow over the airfoil has higher speed than the flow under it, which is required to produce lifting. This is not observed, however, in the regular-grid simulation.

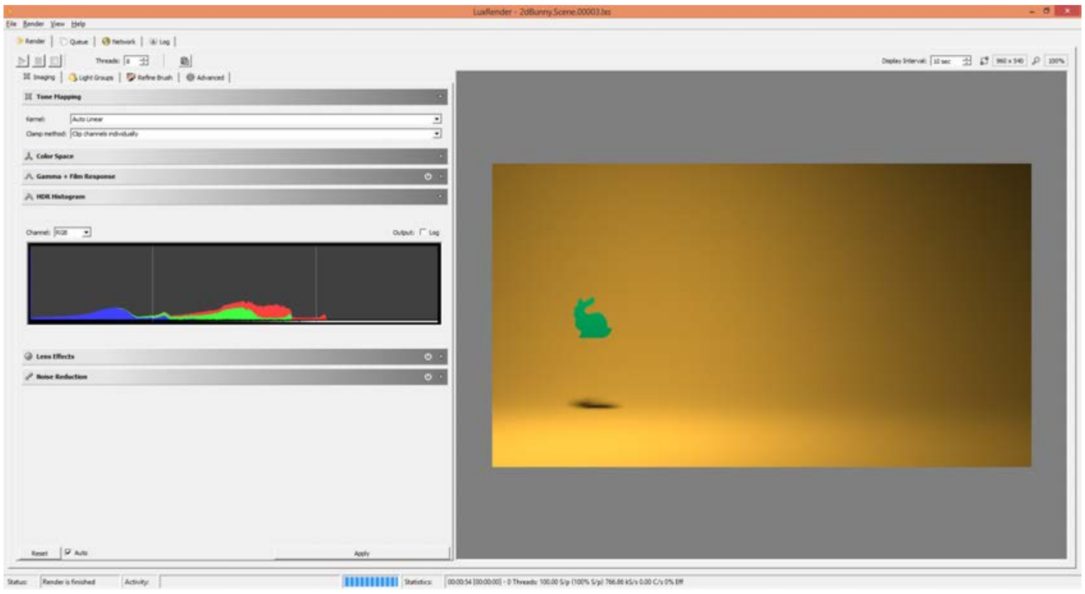


Figure 5.3: Animation pipeline: setup of the scene on Luxrender.

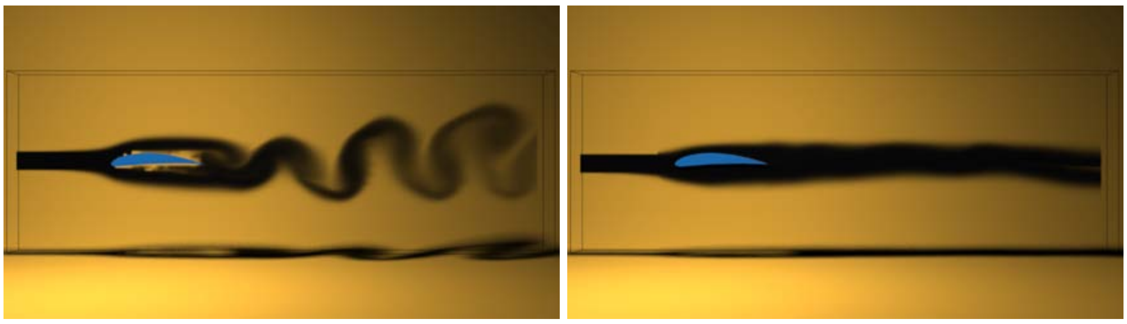


Figure 5.4: Flow simulation using a cross section of a horizontally-oriented airfoil using: a regular grid (left), and our overlapping curvilinear grid approach (right). The use of a regular grid incorrectly leads to the appearance of high-vorticity regions behind the airfoil. In contrast, our approach produces a well-behaved air flow.

Figure 5.6 provides another comparison of our method and the use of regular grids. The regular discretization of the bunny profile lends to incorrect simulation results. Our approach generates correct ones, despite the shapes of the individual cells in the bunny's curvilinear grid be highly irregular.

Figure 5.7 compares our approach (right) with the use of a regular grid (left) using two advection techniques: semi-Lagrangian (top row) and a modified MacCormack method (SELLE et al., 2008) (bottom row). In both cases, our approach produces more dense vortices. The differences in the results produced by our method and by the regular-grid one are due to the use of improper boundary conditions in the regular-grid solution.

Figure 5.9 and Figure 5.10 show various frames of a 3-D smoke simulation in a wind tunnel for visual inspection of a car's aerodynamics properties. Note how the use of appropriate boundary conditions lends to a smooth flow over the car, as well as to the formation of vortices behind it, due to changes in the pressure field. A cross section of the pressure field for a given time step of another simulation is shown in Figure 5.8. In this color-coded representation, red means high pressure. The image on the left depicts the simulation result obtained with a regular grid. Note how the discretization of the car

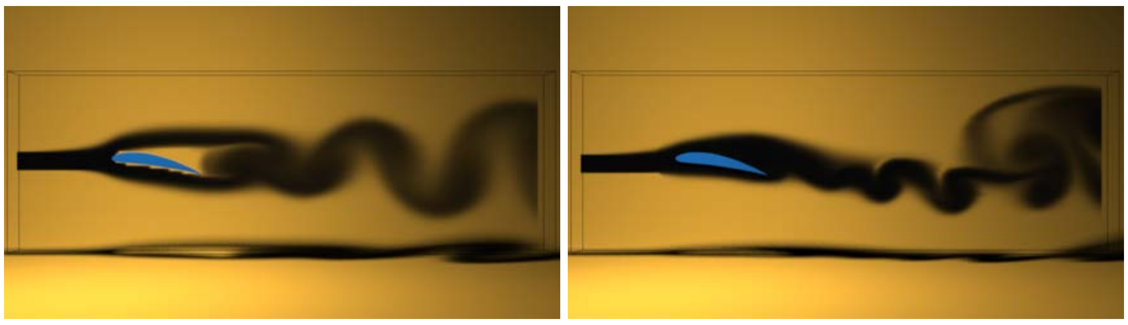


Figure 5.5: Comparison similar to the one shown in Figure 5.4, but orienting the cross section of the air foil at 10 degrees. The use of a regular grid produces a clearly incorrect result (left). Ou method generates a correct simulation (right).

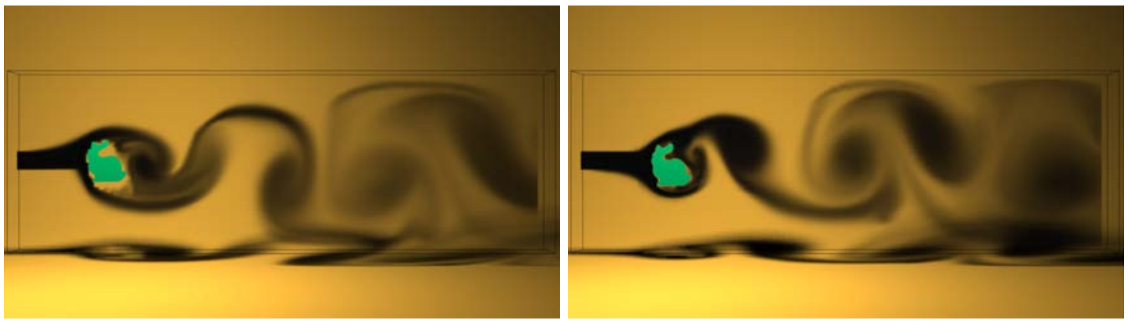


Figure 5.6: 2-D simulation of the bunny shape using a regular grid (left) and our approach (right).

surface incorrectly lends to coarse representation of the pressure field. The pressure field computed with our approach is shown on the right and provides a much more detailed result.

Table 1 compares the performance of our method with the regular-grid approach. The reported times correspond to one simulation step. Measurements were performed on an Intel core i7-2600 3.40 GHZ CPU with 8 GB of RAM and a GeForce GTX 460 with 1 GB of RAM. For the 2-D examples, both techniques use multigrid implementations. The numbers show that the performance of our approach is comparable to the regular grid approach. The quality of our results, on the other hand, is clearly superior.

5.1.1 Variation of the Timestep

The CFL condition is non-uniform for curvilinear grids, due non-regular grid cell spacings that may appear with grid deformations. We generated a single-block curvilinear grid (Figure 5.11) composed by two semi-circular grids to test the behavior of our solution

Example	**Total time (s)**		**Number of cells**	
	Regular	Curvilinear	Regular	Curvilinear
Circular obstacle 2-D	0.033	0.057	20,667	19,206
Bunny obstacle 2-D	0.032	0.048	20,667	17,061
Bunny & Circular 2-D	0.052	0.066	20,667	23,463
Wind tunnel 3-D	3.242	3.654	1,365,525	1,365,525

Table 5.1: Performance comparison between our approach and the regular-grid one.

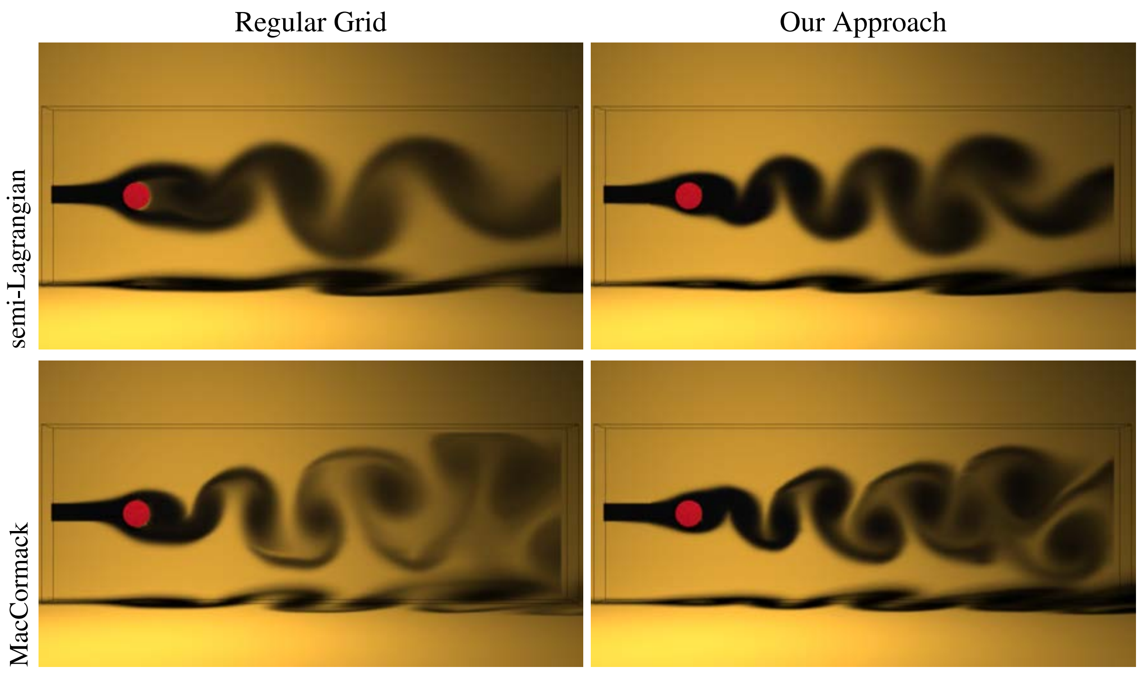


Figure 5.7: Results produced with regular grids (left) and our approach (right) using the semi-Lagrangian (top) and a modified MacCormack method (bottom) for the advection phase.

when higher time-steps are used. For this grid, the local CFL $\Delta t/dx$, with $\Delta t = 0.05$, varies along grid cell sizes: the minimum CFL is 3.9370 while the maximum is 7.8125.

When we use a bigger time-step without ensuring that the local CFL is one (Section 3.4) on curvilinear grids, artifacts tend to appear in regions with increased grid deformation (Figure 5.12 (top)). However, if the multi-step adaptive Runge-Kutta path integrator is used, these artifacts vanish completely, as one can see in Figure 5.12 (bottom). This method also improves the accuracy of the simulation on regular grids.

The velocity dissipation of the semi-Lagrangian algorithm decreases with the cells spacing size (MOLEMAKER et al., 2008). This dissipation can be re-interpreted as viscosity, and increased viscosity induces, to some extent, more vorticity to the flow (ELCOTT et al., 2007; SELLE et al., 2008). On curvilinear grids, a non-uniform artificial dissipation is present on the flow, since cell sizes and spacing varies along the domain.

When we increased the timestep Δt in our simulations, the experimental results shows that the non-uniform artificial dissipation often induces a more complex flow near obstacles. The simulation on curvilinear grids tends to "stick" vortices into objects' boundaries (Figure 5.14), without having disappearing features shown on regular grids (Figure 5.13).

This chapter presented some of the results obtained with our method. We could conclude that our method is able to maintain interesting properties of the fluid flow, such as conservation of aerodynamical properties (Figure 5.4 and Figure 5.5), ability to handle complex grid configurations (Figure 5.6), correct representation of boundary conditions and denser vortices (Figure 5.7), complex 3-D flows (Figure 5.9 and Figure 5.10) and more precise pressure fields (Figure 5.8).

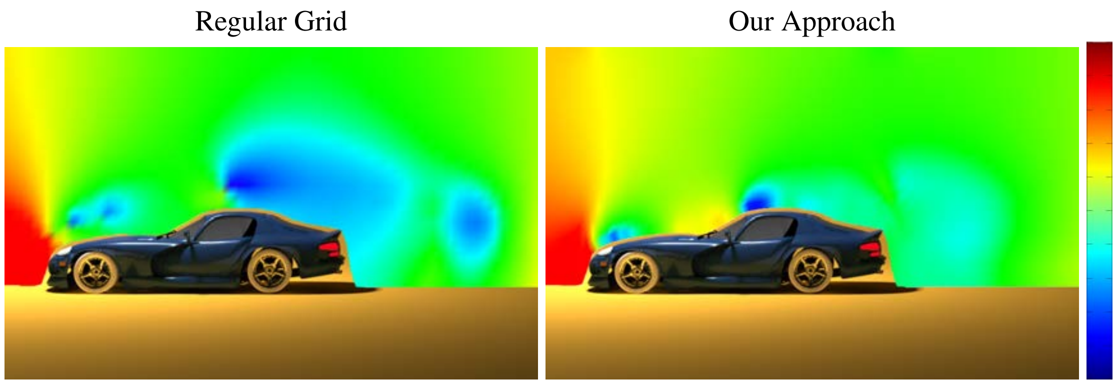


Figure 5.8: Cross sections of the pressure fields for one time step of a 3-D simulation using a regular grid (left) and our approach (right). Red means high pressure.

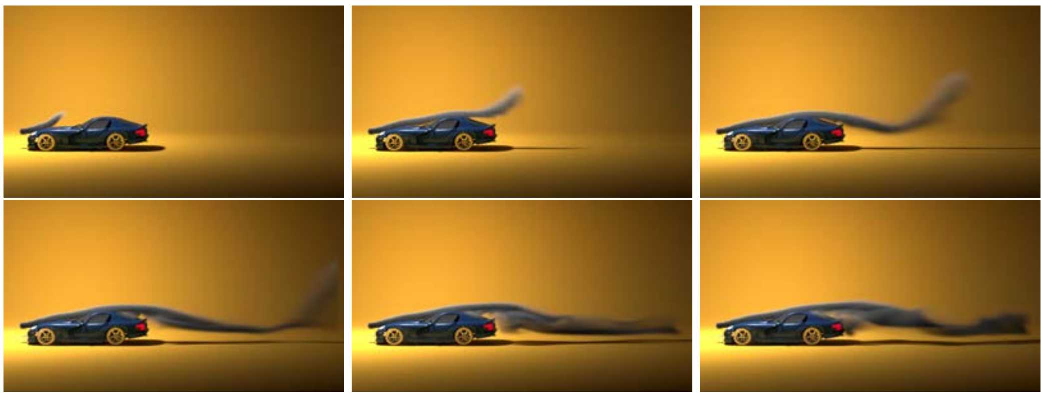


Figure 5.9: The frame sequence (side view) is shown for the 3-D simulation of a car in wind tunnel.

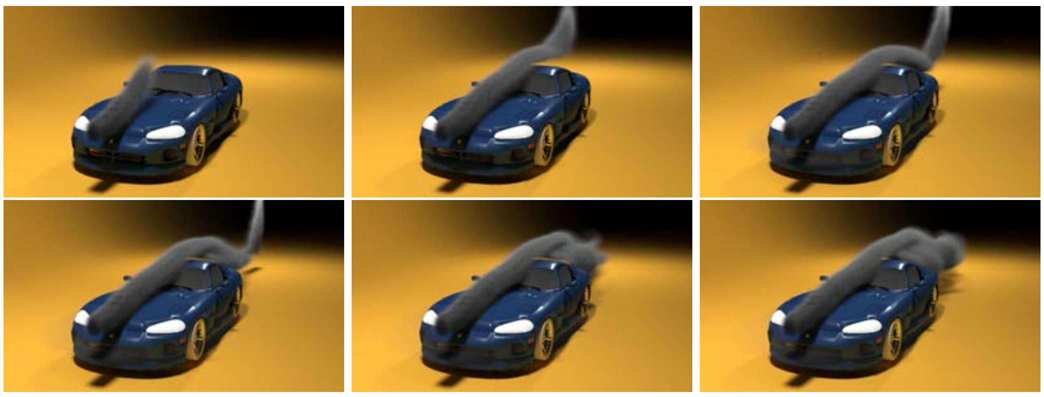


Figure 5.10: The frame sequence (front view) is shown for the 3-D simulation of a car in wind tunnel.

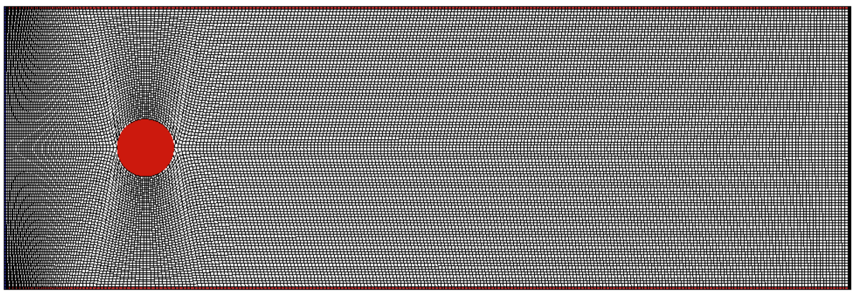


Figure 5.11: 2-D curvilinear non-regular grid composed of two semi-circular patches.

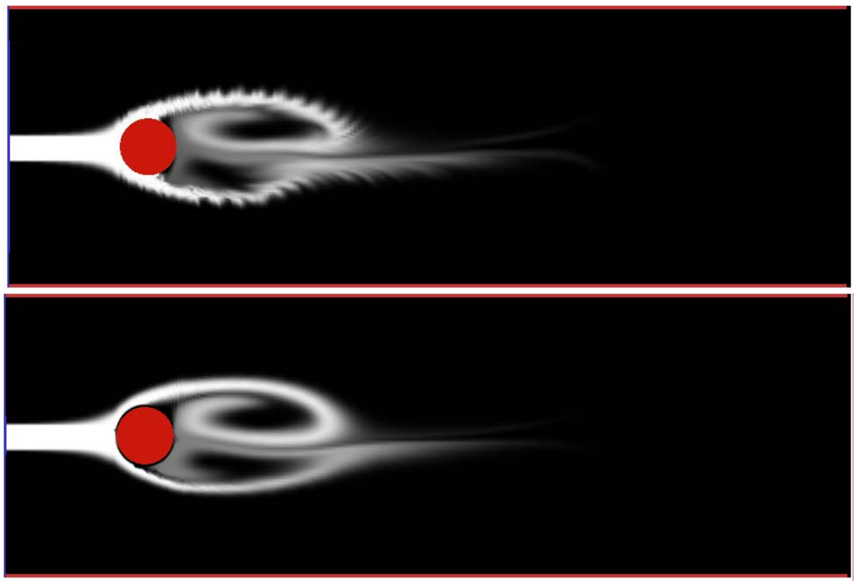


Figure 5.12: 2-D curvilinear grid simulation on a curvilinear grid with timestep $\Delta t = 0.05$. (top): advection using a simple Runge-Kutta method. (bottom) advection using an adaptive Runge-Kutta method.

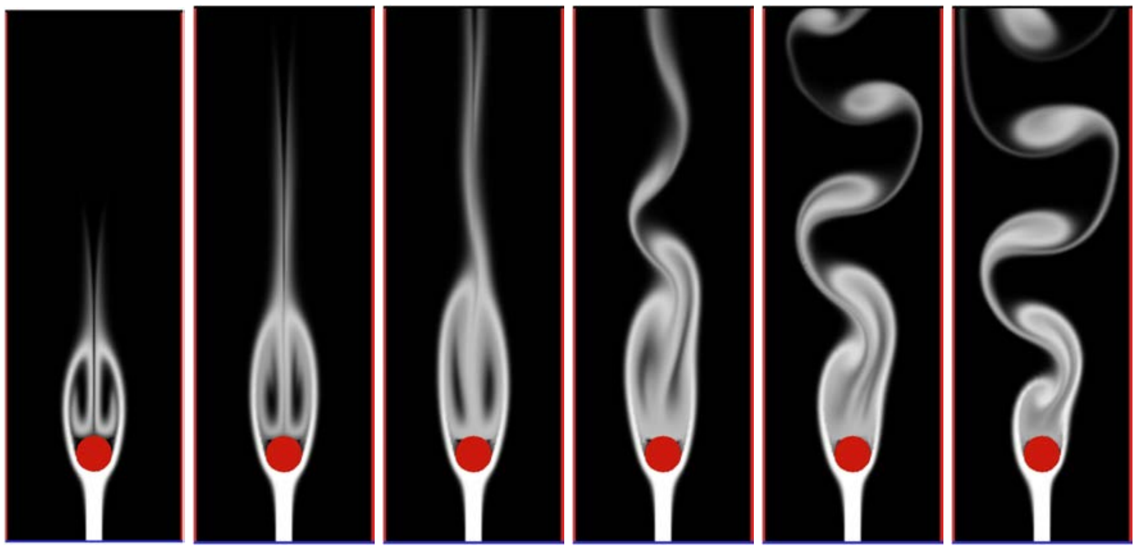


Figure 5.13: 2-D Regular grid simulation of a flow in a channel with an obstacle (timestep $\Delta t = 0.05$). Notice that the vortices tend to be "disconnected" from the obstacle. The vortices also take more time to be formed, when comparing to Figure 5.14.

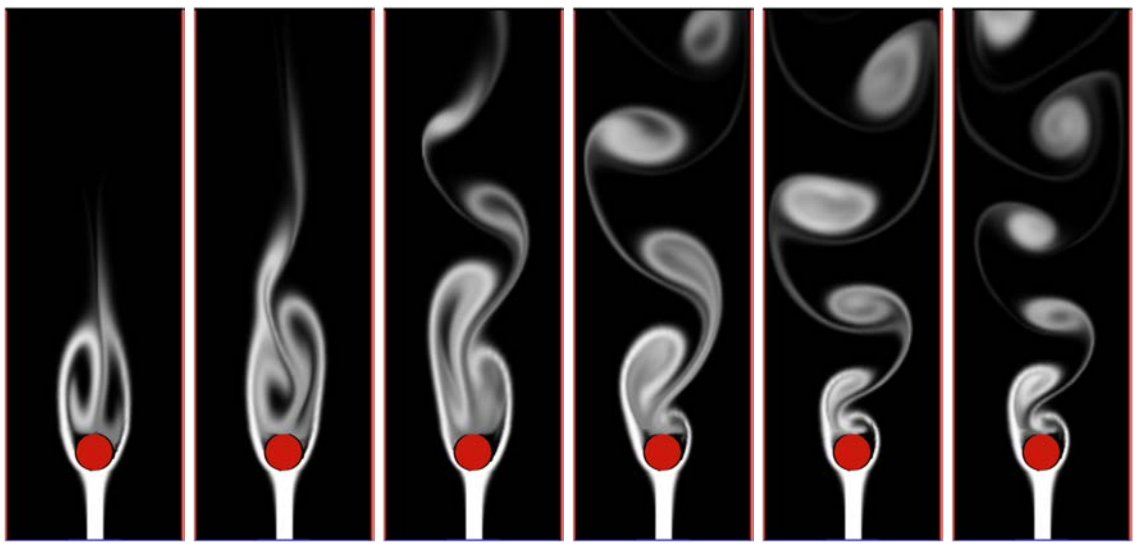


Figure 5.14: 2-D Curvilinear grid simulation of a flow in a channel with an obstacle (timestep $\Delta t = 0.05$). Notice that the vortices tend to "stick" behind the obstacle.

6 CONCLUSIONS AND FUTURE WORK

This thesis presented an efficient approach for creating high-quality flow simulations based on curvilinear grids. Our method is based on techniques originally developed for CFD, and has several desirable properties, including: (i) it allows good definition of boundary conditions, lending to realistic simulations and animations; and (ii) since curvilinear grids are topologically equivalent to regular ones, the cost of the flow solver is nearly identical to the one required to operate on regular grids. Thus, we obtain high-quality results with a relatively low increment in computational cost.

In Chapter 2 we presented a study of the state-of-art fluid animation methods used in computer graphics. The Lagrangian and Eulerian viewpoints were presented, along with the most significant works described so far. Moreover, we discussed the methods proposed by CG researchers to improve the quality of the flow near obstacles and to support local refinement in regular grids. Unstructured grids appeared as an alternate solution for the space discretization of Eulerian simulations. Finally, an outline of the emerging curvilinear grid techniques for animating flows was presented.

Chapter 3 presented our new method for solving the Navier-Stokes equations in curvilinear grids. It enforces mass and momentum conservation by the projection method, subdividing the solution in three steps: *advection*, *pressure solving* and *velocity projection*. On the advection phase, we transformed the physical grid into a computational one, using the velocity expressed in an arbitrary basis to update the trajectory of the particle that calculates the intermediary velocity field. The pressure solving step was formulated based on the integral form of the equations, maintaining the simplicity of the linear system of equations. The velocity projection step was designed to work with the traditional Cartesian staggered-grid variable arrangement. Our projection step is faster than the traditional solution used in CFD, based on D'Yakunov's method (CONCUS; GOLUB, 1972). Finally, we presented our domain decomposition method, which enables a larger applicability of our technique.

Chapter 4 presented the details of the multigrid method for solving linear system of equations on curvilinear grids. Our multigrid approach combines features for several existing techniques, yielding a simple and efficient implementation. We also presented our grid generation method. Since we enable overlapping-grid configurations, our grid generation process is simple and robust. Chapter 5 discussed implementation details and some results obtained with the proposed method. We have demonstrated the effectiveness of our approach through a series of 2-D and 3-D simulations and animations, comparing our method with traditional regular grid approaches.

Our technique could be used to improve the method by Zhu et al. (ZHU; YANG; FAN, 2010) of combining SPH to Eulerian grids. As additional directions for future work, our technique can be extended to handle dynamic environments, supporting rotations and translations, such as (DOBASHI et al., 2008). Moreover, we believe that adapting the techniques described in (LENTINE; ZHENG; FEDKIW, 2010) or (PFAFF et al., 2010) to work with curvilinear grids would greatly enhance the visual aspects of the flows handled by these techniques.

REFERENCES

ABRAMOWITZ, M.; STEGUN, I. A. **Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables**. New York: Dover, 1972.

BANDRINGA, H. **Immersed boundary methods**. 2010. Dissertação (Mestrado em Ciência da Computação) — University of Groningen.

BARR, A. H. Global and local deformations of solid primitives. **SIGGRAPH Comput. Graph.**, New York, NY, USA, v.18, p.21–30, Jan 1984.

BARROSO, V. B. R. B.; CELES, W. Fluid Animation on Arbitrarily-Shaped Structured Grids. In: SBC - PROCEEDINGS OF SBGAMES, 2011. **...** [S.l.: s.n.], 2011.

BATTY, C.; BERTAILS, F.; BRIDSON, R. A fast variational framework for accurate solid-fluid coupling. **ACM Trans. Graph.**, New York, NY, USA, v.26, n.3, p.100, 2007.

BECKER, M.; TESCHNER, M. Weakly compressible SPH for free surface flows. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2007., 2007, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2007. p.209–217.

BELL, N.; GARLAND, M. **CUSP - Generic Parallel Algorithms for Sparse Matrix and Graph Computations.** http://code.google.com/p/cusp-library/. Last access, Nov. 2012.

BELL, N.; HOBEROCK, J. **Thrust - Code at the speed of light.** http://code.google.com/p/thrust/. Last access, Nov. 2012.

BIRD R.B., S. W. E.; LIGHTFOOT, E. N. **Transport phenomena**. New York: Wiley, 1962.

BRANDT, A. Multi-Level Adaptive Solutions to Boundary-Value Problems. **Mathematics of Computation**, [S.l.], v.31, n.138, p.333–390, Apr. 1977.

BRIGGS, W. L.; HENSON, V. E.; MCCORMICK, S. F. **A multigrid tutorial**: second edition. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.

BUTCHER, J. **The numerical analysis of ordinary differential equations**: runge-kutta and general linear methods. New York: Wiler-Interscience, 1982.

CHENTANEZ, N. et al. Liquid simulation on lattice-based tetrahedral meshes. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2007., 2007, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2007. p.219–228. (SCA ’07).

CHENTANEZ, N.; MULLER, M. Real-time Eulerian water simulation using a restricted tall cell grid. **ACM Trans. Graph.**, New York, NY, USA, v.30, n.4, p.82:1–82:10, July 2011.

CHENTANEZ, N.; MULLER, M. A multigrid fluid pressure solver handling separating solid boundary conditions. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2011., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p.83–90. (SCA ’11).

CHESSHIRE, G.; HENSHAW, W. Composite overlapping meshes for the solution of partial differential equations. **Journal of Computational Physics**, [S.l.], v.90, n.1, p.1 – 64, 1990.

CHORIN, A. J. Numerical Solutions of the Navier-Stokes Equations. **Mathematics of computation**, [S.l.], v.22, n.104, p.745–762, October 1968.

COHEN, J. M.; TARIQ, S.; GREEN, S. Interactive fluid-particle simulation using translating Eulerian grids. In: ACM SIGGRAPH I3D, 2010. **Proceedings...** ACM, 2010. p.15–22.

CONCUS, P.; GOLUB, G. H. **Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations.** Stanford, CA, USA: [s.n.], 1972.

CRANE, K.; LLAMAS, I.; TARIQ, S. Chapter 30. Real-Time Simulation and Rendering of 3D Fluids. NGUYEN, H. **GPU Gems 3**. [S.l.]: Addison-Wesley Professional, 2007.

DOBASHI, Y. et al. A Fast Simulation Method Using Overlapping Grids for Interactions between Smoke and Rigid Objects. **Computer Graphics Forum**, [S.l.], v.27, n.2, p.477–486, 2008.

ELCOTT, S. et al. Stable, circulation-preserving, simplicial fluids. **ACM Trans. Graph.**, New York, NY, USA, v.26, n.1, p.4, 2007.

ENRIGHT, D.; MARSCHNER, S.; FEDKIW, R. Animation and rendering of complex water surfaces. **ACM Trans. Graph.**, New York, NY, USA, v.21, n.3, p.736–744, 2002.

FEDKIW, R.; STAM, J.; JENSEN, H. W. Visual simulation of smoke. In: SIGGRAPH’01, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.15–22.

FELDMAN, B. E. et al. Fluids in deforming meshes. In: ACM SIGGRAPH/EUROGRAPHICS SCA, 2005., 2005. **Proceedings...** [S.l.: s.n.], 2005. p.255–259.

FELDMAN, B. E.; O’BRIEN, J. F.; ARIKAN, O. Animating suspended particle explosions. **ACM Trans. Graph.**, New York, NY, USA, v.22, n.3, p.708–715, July 2003.

FELDMAN, B. E.; O’BRIEN, J. F.; KLINGNER, B. M. Animating gases with hybrid meshes. **ACM Trans. Graph.**, New York, NY, USA, v.24, n.3, p.904–909, 2005.

FERZIGER, J. H.; PERIC, M. **Computational methods for fluid dynamics; 2nd ed.** Berlin: Springer, 1999.

FOSTER, N.; FEDKIW, R. Practical animation of liquids. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 28., 2001, New York, NY, USA. **Proceedings...** ACM, 2001. p.23–30. (SIGGRAPH '01).

FOSTER, N.; METAXAS, D. Modeling the motion of a hot, turbulent gas. In: SIGGRAPH'97, 1997. **Proceedings...** [S.l.: s.n.], 1997. p.181–188.

FOX, R.; MCDONALD, A. **Introduction to fluid mechanics**. New York: Wiley, 1982.

HARLOW, F. H.; WELCH, J. E. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. **Physics of Fluids**, [S.l.], v.8, n.12, p.2182–2189, 1965.

HE, P. et al. Multigrid calculation of fluid flows in complex 3D geometries using curvilinear grids. **Computers Fluids**, [S.l.], v.25, n.4, p.395 – 419, 1996.

HENSHAW, W. Adaptive Mesh Refinement on Overlapping Grids. In: PLEWA, T.; LINDE, T.; GREGORY WEIRS, V. (Ed.). **Adaptive Mesh Refinement - Theory and Applications**. [S.l.]: Springer Berlin Heidelberg, 2005. p.59–71. (Lecture Notes in Computational Science and Engineering, v.41).

HENSHAW, W. D. On Multigrid for Overlapping Grids. **SIAM J. Sci. Comput.**, Philadelphia, PA, USA, v.26, n.5, p.1547–1572, May 2005.

HINATSU, M.; FERZIGER, J. H. Numerical computation of unsteady incompressible flow in complex geometry using a composite multigrid technique. **International Journal for Numerical Methods in Fluids**, [S.l.], v.13, n.8, p.971–997, 1991.

HOLLY, F. M. J.; PREISSMAN, A. Accurate Calculation of Transport in Two Dimensions. **Journal of Hydraulic Engineering**, [S.l.], v.103, n.11, p.1259–1277, November 1977.

HONG, J.-M.; SHINAR, T.; FEDKIW, R. Wrinkled flames and cellular patterns. **ACM Trans. Graph.**, New York, NY, USA, v.26, n.3, July 2007.

HOUSTON, B.; BOND, C.; WIEBE, M. A unified approach for modeling complex occlusions in fluid simulations. In: SIGGRAPH 2003 CONFERENCE ON SKETCHES & APPLICATIONS, 2003. **Proceedings...** ACM Press, 2003.

HOUSTON, B. et al. Hierarchical RLE level set: a compact and versatile deformable surface representation. **ACM Trans. Graph.**, New York, NY, USA, v.25, n.1, p.151–175, Jan. 2006.

IGES. **Initial Graphics Exchange Specification. IGES 5.3**. http://www.uspro.org/documents/IGES5-3_forDownload.pdf. Last access, Nov. 2012.

IRVING, G. et al. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In: ACM SIGGRAPH 2006 PAPERS, 2006, New York, NY, USA. **...** ACM, 2006. p.805–811. (SIGGRAPH '06).

ISERLES, A. **A First Course in the Numerical Analysis of Differential Equations**. Cambridge: Cambridge University Press, 1996.

KARPIK, S. R.; CROCKETT, S. R. Semi-Lagrangian Algorithm for Two-Dimensional Advection-Diffusion Equation on Curvilinear Coordinate Meshes. **Journal of Hydraulic Engineering**, [S.l.], v.123, p.389–401, May 1997.

KIM, D.; CHOI, H. A second-order time-accurate finite volume method for unsteady incompressible flow on hybrid unstructured grids. **J. Comput. Phys.**, San Diego, CA, USA, v.162, p.411–428, 2000.

KIM, D.; SONG, O.-y.; KO, H.-S. A Semi-Lagrangian CIP Fluid Solver without Dimensional Splitting. **Computer Graphics Forum**, [S.l.], v.27, n.2, p.467–475, 2008.

KIM, D.; SONG, O.-y.; KO, H.-S. Stretching and wiggling liquids. In: ACM SIGGRAPH ASIA 2009 PAPERS, 2009, New York, NY, USA. **...** ACM, 2009. p.120:1–120:7. (SIGGRAPH Asia '09).

KLINGNER, B. M. et al. Fluid animation with dynamic meshes. **ACM Trans. Graph.**, New York, NY, USA, v.25, n.3, p.820–825, 2006.

KOSHIZUKA, S.; OKA, Y.; KONDO, S. A staggered differencing technique on boundary-ditted curvilinear grids for incompressible flows along curvilinear or slant walls. **Computational Mechanics**, [S.l.], v.7, p.123–136, 1990.

LENTINE, M.; ZHENG, W.; FEDKIW, R. A novel algorithm for incompressible flow using only a coarse grid projection. **ACM Trans. Graph.**, New York, NY, USA, v.29, p.114:1–114:9, July 2010.

LOSASSO, F. et al. Multiple interacting liquids. In: ACM SIGGRAPH 2006 PAPERS, 2006, New York, NY, USA. **...** ACM, 2006. p.812–819. (SIGGRAPH '06).

LOSASSO, F.; FEDKIW, R.; OSHER, S. Spatially adaptive techniques for level set methods and incompressible flow. **Computers and Fluids**, [S.l.], v.35, p.2006, 2005.

LOSASSO, F.; GIBOU, F.; FEDKIW, R. Simulating water and smoke with an octree data structure. **ACM Trans. Graph.**, New York, NY, USA, v.23, n.3, p.457–462, Aug. 2004.

LUXRENDER. **Luxrender - GPL Based renderer.** http://www.luxrender.net/. Last access, Nov. 2012.

MCADAMS, A.; SIFAKIS, E.; TERAN, J. A parallel multigrid Poisson solver for fluids simulation on large grids. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2010., 2010, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2010. p.65–74. (SCA '10).

MOLEMAKER, J. et al. Low viscosity flow simulations for animation. In: ACM SIGGRAPH/EUROGRAPHICS SCA, 2008., 2008. **Proceedings...** [S.l.: s.n.], 2008. p.9–18.

MOSTOW, D. J. **Terminator 3**: rise of the machines. 2003.

MULLER, M.; CHARYPAR, D.; GROSS, M. Particle-based fluid simulation for interactive applications. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2003., 2003, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2003. p.154–159.

MULLER, M. et al. Particle-based fluid-fluid interaction. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2005., 2005, New York, NY, USA. **Proceedings...** ACM, 2005. p.237–244.

NGUYEN, D. Q.; FEDKIW, R.; JENSEN, H. W. Physically based modeling and animation of fire. **ACM Trans. Graph.**, New York, NY, USA, v.21, n.3, p.721–728, July 2002.

NVIDIA. **NVIDIA's PhysX SDK Download.** https://developer.nvidia.com/physx-downloads. Last access, Nov. 2012.

NVIDIA. **NVIDIA's Cg Toolkit download.** https://developer.nvidia.com/cg-toolkit. Last access, Nov. 2012.

PARK, S. I.; KIM, M. J. Vortex fluid for gaseous phenomena. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2005., 2005, New York, NY, USA. **Proceedings...** ACM, 2005. p.261–270. (SCA '05).

PETERSEN, D. W. **Poseidon**. 2006.

PFAFF, T. et al. Scalable fluid simulation using anisotropic turbulence particles. In: ACM SIGGRAPH ASIA 2010 PAPERS, 2010, New York, NY, USA. **...** ACM, 2010. p.174:1–174:8. (SIGGRAPH ASIA '10).

PHARR, M.; HUMPHREYS, G. **Physically based rendering - From theory to implementation.** http://www.pbrt.org/. Last access, Nov. 2012.

POINTWISE. **Pointwise Gridgen - Reliable CFD Meshing Software.** http://www.pointwise.com/gridgen/. Last access, Nov. 2012.

RASMUSSEN, N. et al. Directable photorealistic liquids. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2004., 2004, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2004. p.193–202. (SCA '04).

ROBERT, A. A stable numerical integration scheme for the primitive meteorological equations. **Atmosphere-Ocean**, [S.l.], v.19, n.1, p.35–46, 1981.

ROBINSON-MOSHER, A.; ENGLISH, R. E.; FEDKIW, R. Accurate tangential velocities for solid fluid coupling. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 2009., 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p.227–236. (SCA '09).

ROBINSON-MOSHER, A. et al. Two-way coupling of fluids to rigid and deformable solids and shells. **ACM Trans. Graph.**, New York, NY, USA, v.27, n.3, p.46:1–46:9, Aug. 2008.

ROBLE, D.; ZAFAR, N. b.; FALT, H. Cartesian grid fluid simulation with irregular boundary voxels. In: ACM SIGGRAPH 2005 SKETCHES, 2005, New York, NY, USA. **...** ACM, 2005. (SIGGRAPH '05).

SCHECHTER, H.; BRIDSON, R. Ghost SPH for animating water. **ACM Trans. Graph.**, New York, NY, USA, v.31, n.4, p.61:1–61:8, July 2012.

SEDOV, L. **A course on continuum mechanics**. Groningen: Walters-Noordhoft, 1972.

SELLE, A. et al. An Unconditionally Stable MacCormack Method. **J. Sci. Comput.**, New York, NY, USA, v.35, p.350–371, June 2008.

SELLE, A.; RASMUSSEN, N.; FEDKIW, R. A vortex particle method for smoke, water and explosions. In: ACM SIGGRAPH 2005 PAPERS, 2005, New York, NY, USA. **...** ACM, 2005. p.910–914. (SIGGRAPH '05).

SHYY, W.; VU, T. C. On the adoption of velocity variable and grid system for fluid flow computation in curvilinear coordinates. **Journal of Computational Physics**, [S.l.], v.92, n.1, p.82 – 105, 1991.

SOLENTHALER, B.; PAJAROLA, R. Predictive-corrective incompressible SPH. In: ACM SIGGRAPH PAPERS, 2009., 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p.1–6.

STAM, J. Stable fluids. In: SIGGRAPH'99, 1999. **Proceedings...** [S.l.: s.n.], 1999. p.121–128.

STAM, J. Flows on surfaces of arbitrary topology. **ACM Trans. Graph.**, New York, NY, USA, v.22, n.3, p.724–731, July 2003.

TILCH, R. et al. Combination of Body-Fitted and Embedded Grids for External Vehicle Aerodynamics. In: ENGINEERING COMPUTATIONS 25, NO. 1, 2008. **Proceedings...** [S.l.: s.n.], 2008. p.280–41.

TRUESDELL, C. **A first course in rational continuum mechanics**. London: Academic Press, 1977.

WENDT, J. D. et al. Finite volume flow simulations on arbitrary domains. **Graph. Models**, San Diego, CA, USA, v.69, p.19–32, January 2007.

WESSELING, P.; SEGAL, A.; KASSELS, C. G. M. Computing flows on general three-dimensional nonsmooth staggered grids. **J. Comput. Phys.**, San Diego, CA, USA, v.149, p.333–362, March 1999.

WYMAN, N. **State of the Art in Grid Generation. CFD Review**. http://www.cfdreview.com/article.pl?sid=01/04/28/2131215. Last access, Sept. 2012.

ZANG, Y.; STREET, R. L.; KOSEFF, J. R. A Non-staggered Grid, Fractional Step Method for Time-Dependent Incompressible Navier-Stokes Equations in Curvilinear Coordinates. **Journal of Computational Physics**, [S.l.], v.114, n.1, p.18 – 33, 1994.

ZHU, B.; YANG, X.; FAN, Y. Creating and Preserving Vortical Details in SPH Fluid. **Computer Graphics Forum**, [S.l.], v.29, n.7, p.2207–2214, 2010.

APPENDIX A MATHEMATICAL CONCEPTS

This Appendix introduces some mathematical concepts and notations that are relevant for the technique developed in this thesis.

A.1 Conservation Equations

This section provides a brief review of the momentum and mass conservation equations for fluid flow. It is based on classic Newtonian continuum mechanics and conservation laws. Continuum mechanics deals with the analysis of the kinematics and the mechanical behavior of materials modeled as a continuous mass rather than as discrete particles.

Conservation laws state that a particular measurable property of an isolated physical system does not change as the system evolves. The following demonstrations are based on (FERZIGER; PERIC, 1999). To formulate that is no creation or destruction of mass within a system (mass conservation law), one can write

$$\frac{\mathrm{d}m}{\mathrm{d}t} = 0. \tag{A.1}$$

Another example can be observed in Newton's second law of motion (momentum conservation law), which states that momentum is the sum of all forces acting on the system

$$\frac{\mathrm{d}(m\vec{u})}{\mathrm{d}t} = \sum \vec{f}. \tag{A.2}$$

Conservation laws can be derived by considering a given quantity of matter or *control mass* (CM) and its *extensive properties*, such as mass, momentum and energy. This approach is successfully adopted in the study of dynamics of solid bodies. However, for the dynamics of fluid flows it is difficult to follow a parcel of matter and it is more convenient to define the conservation laws within a *control volume* (CV).

To use the mass and momentum conservation laws into within a control volume, the fundamental variables will be considered as *intensive* rather than extensive properties; the former are independent of the amount of matter considered. Examples are density ρ (mass per unit volume) and velocity $\vec{u}$ (momentum per unit volume). If ϕ is any conserved intensive property the relation to its corresponding extensive property Φ is expressed as

$$\Phi = \int_{\Omega_{CM}} \rho\phi \, \mathrm{d}\Omega \tag{A.3}$$

Using this definition, the control volume equation or Reynolds transport theorem for a fixed CV is defined by

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi = \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CM}} \rho\phi \,\mathrm{d}\Omega = \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CV}} \rho\phi \,\mathrm{d}\Omega + \int_{S_{CV}} \rho\phi\vec{u}\cdot\vec{n}dS, \tag{A.4}$$

where Ω_{CV} is the CV volume, S_{CV} is the surface enclosing the control volume, $\vec{n}$ the orthogonal unit vector relative to the control volume surface directed outwards and $\vec{u}$ is the fluid velocity. This equation states that a variation of Φ is given by the sum of the variation of ϕ integrated on the control volume (first term of Eq. (A.4)), and the ratio of which ϕ is entering or leaving the CV (second term of Eq. (A.4). The detailed derivation of Eq. (A.4) is given in various textbooks (BIRD R.B.; LIGHTFOOT, 1962; FOX; MCDONALD, 1982) and will not be repeated here. The mass and momentum conservation equations in their integral forms will be derived from Eq. (A.4).

A.1.1 Mass Conservation

The mass conservation law states that there is no creation or destruction of mass within a system (Eq. (A.1)). Considering this, the integral form of the mass conservation equation follows directly from the control volume Eq. (A.4), by setting its left-hand side to zero and making $\phi = 1$:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho \;\mathrm{d}\Omega + \int_{S} \rho\vec{u}\cdot\vec{n} \;\mathrm{d}S = 0. \tag{A.5}$$

Applying the Gauss' divergence theorem to the convection term, we can transform the surface integral into a volume integral

$$\frac{\partial}{\partial t}\int_{\Omega} \rho \;\mathrm{d}\Omega + \int_{\Omega} \rho\vec{u} \;\mathrm{d}\Omega = 0. \tag{A.6}$$

To obtain the differential coordinate-free form of the continuity equation, we set $\Omega \to 0$, yielding

$$\frac{\partial\rho}{\partial t} + \mathrm{div}\,(\rho\vec{u}) = 0. \tag{A.7}$$

With the discretization of the divergence operator we can transform this form to a specific coordinate system (SEDOV, 1972; TRUESDELL, 1977). If we assume that the fluids are incompressible, *i.e.*, there is no variation on the density ρ, Eq. (A.6) and Eq. (A.7) become

$$\int_{S} \vec{u}\cdot\vec{n} \;\mathrm{d}S = 0, \tag{A.8}$$

$$\nabla\cdot\vec{u} = 0. \tag{A.9}$$

A.1.2 Momentum Conservation

Assuming the same principle used in the mass conservation law, $\phi = \vec{u}$ and the control volume Eq. (A.4), we have for a fixed fluid-containing volume of space:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\vec{u}\,\mathrm{d}\Omega + \int_{S} \rho\vec{u}\vec{u}\cdot\vec{n}\,\mathrm{d}S = \sum\vec{f}. \tag{A.10}$$

The forces acting on a control volume can be of two distinct types: surface forces (pressure, normal and shear stresses, surface tension, etc), and body forces (gravity, centrifugal and Coriolis forces, etc). Surface forces act directly on a CV surface, *e.g.*, the shear force due to wind blowing above the ocean. Body forces are forces that act in the whole CV, like gravity. Representing surface forces by $\mathbf{T}$ and the body forces by $\mathbf{b}$, the integral form of the conservation equation for the momentum becomes

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\vec{u}\,\mathrm{d}\Omega + \int_{S}\rho\vec{u}\vec{u}\cdot\vec{n}\,\mathrm{d}S = \int_{S}\mathbf{T}\cdot\vec{n}\,\mathrm{d}S + \int_{\Omega}\rho\mathbf{b}\,\mathrm{d}\Omega. \tag{A.11}$$

For Newtonian fluids, whose the rate of stress (internal forces) and the strain rate (deformation) is linear, the stress tensor $\mathbf{T}$ can be written by

$$\mathbf{T} = -(p + \frac{2}{3}\mu\mathrm{div}\ \vec{u}) + 2\mu\mathbf{D}, \tag{A.12}$$

where $\mathbf{D}$ is

$$\mathbf{D} = \frac{1}{2}[\mathrm{grad}\ \vec{u} + (\mathrm{grad}\ \vec{u})^T]. \tag{A.13}$$

Making use of the index notation, is useful to define the shear stress tensor due to viscous forces as

$$\tau_{ij} = 2\mu\mathbf{D} - \frac{2}{3}\mu\delta_{ij}\mathrm{div}\ \vec{u}, \tag{A.14}$$

where δ_{ij} is the Kroenecker delta.

The differential coordinate free-form of the Eq. (A.11) is given by:

$$\frac{\partial(\rho\vec{u})}{\partial t} + \mathrm{div}\ (\rho\vec{u}\vec{u}) = \mathrm{div}\ \mathbf{T} + \rho\mathbf{b}. \tag{A.15}$$

A.2 Numerical Discretizations

A.2.1 Finite Difference Method

The finite difference method (FDM) is the most popular method for solving the Navier-Stokes equations in computer graphics. Its natural choice relies on the fact that it is the easiest method for simple Cartesian structured grids. With some effort, the FDM method can be used with non-regular structured grids; however, apparently, FDM has never been used with unstructured grids.

The starting point to solve the equations are the differential forms, *e.g.*the momentum conservation the differential form is represented by Eq. (A.15). It can be discretized relative to the *i*th Cartesian component as

$$\frac{\partial(\rho\mathbf{u}_i)}{\partial t} + \mathrm{div}\ (\rho\mathbf{u}_i\vec{u}) = \mathrm{div}\ \mathbf{t}_i + \rho\mathbf{b}_i, \tag{A.16}$$

where

$$\mathbf{t}_i = \mu\mathrm{grad}\ \mathbf{u}_i + \mu(\mathrm{grad}\ \vec{u})^T\cdot\mathbf{i}_i - (p + \frac{2}{3}\mu\mathrm{div}\ \vec{u})\mathbf{i}_i = \tau_{ij} - p\mathbf{i}_i. \tag{A.17}$$

Here $\mathbf{t}_i$ is the *i*th vector from tensor $\mathbf{T}$, $\mathbf{b}_i$ is the *i*th component of the body force vector and $\mathbf{i}_i$ is the Cartesian unit vector in the direction of the coordinate $\mathbf{x}_i$. It is convenient to separate all the stress due to shear forces into a shear tensor (τ_{ij}).

Eq. (A.16) is in strongly conservative form, since all terms have the form of the divergence of a vector or tensor (FERZIGER; PERIC, 1999). The usual form used in finite difference methods is the chain-rule conservative form, that represents the divergent operator in Eq. (A.16) as

$$\text{div } (\rho \mathbf{u}_i \vec{u}) = \mathbf{u}_i \text{ div } (\rho \vec{u}) + \rho \vec{u} \cdot \text{grad } \mathbf{u}_i. \tag{A.18}$$

Thus, using Eq. (A.18) into Eq. (A.15), it follows that:

$$\frac{\partial(\rho \mathbf{u}_i)}{\partial t} + \rho \vec{u} \cdot \text{grad } \mathbf{u}_i = \text{div } \mathbf{t}_i + \rho \mathbf{b}_i. \tag{A.19}$$

Assuming that the only body force that acts in the fluid is the gravity, using the stress tensor and applying the gradient and divergence operators in Cartesian coordinate systems we have

$$\frac{\partial(\rho \mathbf{u}_i)}{\partial t} + \frac{\partial(\rho \mathbf{u}_j \mathbf{u}_i)}{\partial \mathbf{x}_j} = \frac{\partial \tau_{ij}}{\partial \mathbf{x}_j} - \frac{\partial p}{\partial x_i} + \rho g_i. \tag{A.20}$$

Most of finite difference methods use Eq. (A.20) as the starting point. The basic concept of the finite difference method is to consider each grid point as the origin of a local coordinate system. The grid lines of the same family may not intersect, otherwise the same subspace would be mapped by different grid points. With this, the finite difference approximations are borrowed directly from the definition of derivatives:

$$\left(\frac{\partial \phi}{\partial x}\right)_{x_i} = \lim_{\Delta x \to 0} \frac{\phi(x_i + \Delta x) - \phi(x_i)}{\Delta x} \tag{A.21}$$

There are three basic approaches to discretize Eq. (A.21): Taylor Series expansion, polynomial fitting and compact schemes. We are going to present only Taylor Series expansion, since it is the most used method in context of computer graphics. Any continuous differentiable function $\phi(x)$ can, in the the vicinity of x_i, can be expressed as a Taylor Series (ABRAMOWITZ; STEGUN, 1972):

$$\begin{aligned} \phi(x) = \phi(x_i) + (x - x_i)\left(\tfrac{\partial \phi}{\partial x}\right)_i + \tfrac{(x-x_i)^2}{2!}\left(\tfrac{\partial^2 \phi}{\partial x^2}\right)_i + \\ \tfrac{(x-x_i)^3}{3!}\left(\tfrac{\partial^3 \phi}{\partial x^3}\right)_i + ... + \tfrac{(x-x_i)^n}{n!}\left(\tfrac{\partial^n \phi}{\partial x^n}\right)_i + H, \end{aligned} \tag{A.22}$$

where H means "higher order terms". Therefore, we can approximate the 1-D derivative of Eq. (A.21) at x_i using the next point x_{i+1} by

$$\left(\frac{\partial \phi}{\partial x}\right)_{x_i} = \frac{\phi(x_{i+1}) - \phi(x)}{x_{i+1} - x_i} - \frac{x_{i+1} - x_i}{2}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_i + H. \tag{A.23}$$

Another expression can be derived for Eq. (A.21) using the previous point x_{i-1}:

$$\left(\frac{\partial \phi}{\partial x}\right)_{x_i} = \frac{\phi(x_{i+1}) - \phi(x)}{x_{i+1} - x_i} - \frac{x_{i+1} - x_i}{2}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_i + H. \tag{A.24}$$

Still another expression can be obtained by using both x_{i-1} and x_{i+1}:

$$\left(\frac{\partial \phi}{\partial x}\right)_{x_i} = \frac{\phi(x_{i+1}) - \phi(x)_{i-1}}{x_{i+1} - x_{i-1}} - \frac{(x_{i+1} - x_i)^2 - (x_i - x_{i-1})^2}{2(x_{i+1} - x_{i-1}}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_i + H. \tag{A.25}$$

These three different approaches for discretizing Eq. (A.21) are known, respectively, as forward- (FDS), backward- (BDS) and central-difference (CDS) schemes and they are used to compute first derivatives. Analogously, the second derivatives of $\phi(x)$ can be written with these schemes. However, since our method only uses first derivatives, we will not present these discretization schemes here. For further information, wee refer the reader to (FOX; MCDONALD, 1982; FERZIGER; PERIC, 1999).

A.2.2 Finite Volume Methods

The finite volume method (FVM) use the integral form of the conservation equations as its starting point (Eq. (A.11)). The spatial domain is subdivided into a finite number of control volumes (CVs) and the equations are applied to each CV. Since the method only defines the computational volume boundaries, it is suitable for any type of non-regular grids and it is not directly related to any coordinate system. The disadvantage of FVM with respect to FDM is that is difficult to generate higher order approximations. This is due the fact that FVM requires three levels of approximations: interpolation, differentiation and integration.

In order to obtain an algebraic equation for a particular CV, the surface and volume integrals must be approximated. The surface integrals that relate the net flux of a given quantity through a CV boundary is

$$\int_S f \,\mathrm{d}S = \sum_k \int_{S_k} f \,\mathrm{d}S, \tag{A.26}$$

where k denotes the iteration of all CV's surfaces and f is the component of the flux vector in the direction normal to a CV face. The simplest second order approximation to the integral of one surface is the mid-point rule

$$\int_{S_k} f \,\mathrm{d}S = \overline{f_k} S_k \approx f_k S_k, \tag{A.27}$$

where f_k is the value of the function at the surface center and S_k is the surface area. If the value f_k is not known, it can be interpolated from nearby cell face centers or from other locations where the variables are available.

We can also use the mid-point rule to approximate second order volume integrals, yielding

$$\int_\Omega q \,\mathrm{d}\Omega = \overline{q}\Delta\Omega \approx q_p V, \tag{A.28}$$

where q_p is the value of the function at the CV center and V is the CV volume.

A.2.3 Ordinary differential equations

An ordinary differential equation (ODE) is an equation containing a function of one independent variable and its derivatives. In the context of our fluid solver, we will use a first order ODE to integrate a particle position in the space with respect to time. Given an initial condition, this can be expressed by

$$\left(\tfrac{\mathrm{d}x(t)}{\mathrm{d}t}\right)_t = f(t, x(t)) \quad \text{with initial condition} \quad x(t_0) = x^0. \tag{A.29}$$

The basic problem is the find the position x after a short time Δt. The solution obtained can be regarded as a new initial condition, and the solution can be subsequently advanced in time. Integrating Eq. (A.29) we have

$$x^{n+1} - x^n = \int_{t_n}^{t^{n+1}} f(t, x(t)) \,\mathrm{d}t. \tag{A.30}$$

We can integrate Eq. (A.30) using the current value of x^n as initial point with

$$x^{n+1} = x^n + f(t_n, x^n)\,\Delta t, \tag{A.31}$$

which is known as the **explicit** or **forward Euler method**. This method is very simple to implement and is widely used in many physical simulators. We can also approximate the integral of Eq. (A.30) by the final point $x^{(n+1)}$, yielding

$$x^{n+1} = x^n + f(t_{n+1}, x^{n+1})\,\Delta t, \tag{A.32}$$

which is known as the **implicit** or **backward Euler method**. This second method yields a system of linear equations, but is more stable than the first one. Euler methods have the discretization error of order $O(\Delta t)$, which means that the error linearly decreases with Δt (BUTCHER, 1982; ISERLES, 1996).

For the equations that we want to solve, Euler methods are not precise enough. When used in momentum equations they often generate spiraling artifacts which can be seen in the simulation. Therefore, we adopted a higher-order **Runge-Kutta method**, which evaluates multiple points between t_n and t_{n+1}. This approximation is able to solve Eq. (A.30) with $O(\Delta t^2)$ error. The Runge-Kutta half-step method is represented as

$$\begin{aligned} x^*_{n+\frac{1}{2}} &= x^n + f(t_n, x^n)\,\frac{\Delta t}{2}, \\ x^{n+1} &= x^n + f(t_{n+\frac{1}{2}}, x^*_{n+\frac{1}{2}})\,\Delta t. \end{aligned} \tag{A.33}$$

In this method, an intermediary step is performed to evaluate the velocity at $x^*_{n+\frac{1}{2}}$. Then, this intermediary velocity is used to integrate Eq. (A.30) with a simple explicit Euler method (Eq. (A.31)). The Runge-Kutta half-step method is easy to use and is self-starting - it requires no data other than the initial condition provided by the problem formulation.

A.3 Semi-Lagrangian advection on curvilinear coordinates

In order to show that the velocity in Semi-Lagrangian algorithm works by substituting the transformed velocity in the back-trajectory step, we define the **covariant** basis vectors as

$$\mathbf{g_1} = \frac{\mathbf{g^2} \times \mathbf{g^3}}{\vartheta} \quad \mathbf{g_2} = \frac{\mathbf{g^3} \times \mathbf{g^1}}{\vartheta} \quad \mathbf{g_3} = \frac{\mathbf{g^1} \times \mathbf{g^2}}{\vartheta}, \tag{A.34}$$

with $\vartheta = \mathbf{g^1} \cdot (\mathbf{g^2} \times \mathbf{g^3})$ the volume defined by the three contravariant basis. Assuming that the velocity on the physical domain is given by $\vec{v} = dx/dt$, we can use the chain rule is used to express it on the computational domain in a contravariant basis as

$$\vec{v} = \frac{\partial \xi}{\partial x}\frac{dx}{dt} + \frac{\partial \eta}{\partial y}\frac{dy}{dt} + \frac{\partial \tau}{\partial z}\frac{dz}{dt} = \mathbf{g_1}\frac{dx}{dt} + \mathbf{g_2}\frac{dy}{dt} + \mathbf{g_3}\frac{dz}{dt}. \tag{A.35}$$

Expressing the velocity using the unit covariant basis

$$\mathbf{h_1 u^1 g_{(1)}} + \mathbf{h_2 u^2 g_{(2)}} + \mathbf{h_3 u^3 g_{(3)}} = \mathbf{g_1}\frac{dx}{dt} + \mathbf{g_2}\frac{dy}{dt} + \mathbf{g_3}\frac{dz}{dt}. \tag{A.36}$$

The unit covariant basis can be canceled using the Eq. (3.8), leaving

$$\mathbf{h_1 u^1} + \mathbf{h_2 u^2} + \mathbf{h_3 u^3} = \frac{dx}{dt} + \frac{dy}{dt} + \frac{dz}{dt}. \tag{A.37}$$

The method could be directly evaluated using Eq. (A.37), however it is more convenient to work on contravariant basis. To transform to contravariant basis, the notion of physical vectors is used

$$\mathbf{u_{(i)}} = h_i u^i = \frac{1}{h_i} u_i. \tag{A.38}$$

Using physical vectors to transform Eq. (A.37) into contravariant basis lends to

$$\frac{dx}{dt} = \frac{\mathbf{u_{(1)}}}{\mathbf{h_1}} + \frac{\mathbf{u_{(2)}}}{\mathbf{h_2}} + \frac{\mathbf{u_{(3)}}}{\mathbf{h_3}}. \tag{A.39}$$

A.4 Coupling of pressure and velocity equations

To show that our pressure method is consistent and generates divergence free flows, we substitute Eq. (3.17) in Eq. (3.13), yielding

$$\int_S [\vec{u}^* - \Delta t(\frac{\partial p}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial p}{\partial y}\frac{\partial y}{\partial \eta})\ \mathrm{d}S] \cdot \vec{n} = 0 \tag{A.40}$$

Discretizing the equation for single cartesian oriented components, for the cell (i, j), yields

$$\begin{aligned}
A_{i+\frac{1}{2},j}\left[\vec{u}^*_{i+\frac{1}{2},j} - \Delta t\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta d}\frac{\partial x}{\partial \xi} + \tilde{P}_{i+\frac{1}{2},j}\right)\right] \cdot \vec{n}_{i+\frac{1}{2},j} + \\
A_{i-\frac{1}{2},j}\left[\vec{u}^*_{i-\frac{1}{2},j} - \Delta t\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta d}\frac{\partial x}{\partial \xi} + \tilde{P}_{i-\frac{1}{2},j}\right)\right] \cdot \vec{n}_{i-\frac{1}{2},j} + \\
A_{i,j+\frac{1}{2}}\left[\vec{u}^*_{i,j+\frac{1}{2}} - \Delta t\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta d}\frac{\partial y}{\partial \eta} + \tilde{P}_{i,j+\frac{1}{2}}\right)\right] \cdot \vec{n}_{i,j+\frac{1}{2}} + \\
A_{i,j-\frac{1}{2}}\left[\vec{u}^*_{i,j-\frac{1}{2}} - \Delta t\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta d}\frac{\partial y}{\partial \eta} + \tilde{P}_{i,j-\frac{1}{2}}\right)\right] \cdot \vec{n}_{i,j-\frac{1}{2}} = 0,
\end{aligned} \tag{A.41}$$

where each $\tilde{P}$ is represented by

$$\begin{aligned}
\tilde{P}_{i+\frac{1}{2},j} &= \frac{\tilde{p}_{i+\frac{1}{2},j+\frac{1}{2}} - \tilde{p}_{i+\frac{1}{2},j-\frac{1}{2}}}{\Delta d}\frac{\partial x}{\partial \eta} \\
\tilde{P}_{i-\frac{1}{2},j} &= \frac{\tilde{p}_{i-\frac{1}{2},j+\frac{1}{2}} - \tilde{p}_{i-\frac{1}{2},j-\frac{1}{2}}}{\Delta d}\frac{\partial x}{\partial \eta} \\
\tilde{P}_{i,j+\frac{1}{2}} &= \frac{\tilde{p}_{i+\frac{1}{2},j+\frac{1}{2}} - \tilde{p}_{i-\frac{1}{2},j+\frac{1}{2}}}{\Delta d}\frac{\partial y}{\partial \xi} \\
\tilde{P}_{i,j-\frac{1}{2}} &= \frac{\tilde{p}_{i+\frac{1}{2},j-\frac{1}{2}} - \tilde{p}_{i-\frac{1}{2},j-\frac{1}{2}}}{\Delta d}\frac{\partial y}{\partial \xi}.
\end{aligned} \tag{A.42}$$

In (SHYY; VU, 1991), the authors note that the off-diagonal pressure contributions ($\tilde{P}$) present in Eq. (A.41) can be disregarded in the pressure solving step, without affecting the overall method solution. This simplification can only affect the convergence rate of the solution. Therefore, we discard the ($\tilde{P}$) pressures in the pressure solving system. In 2D, the simplified pressure system is

$$\begin{aligned}
A_{i+\frac{1}{2},j}\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta d}\frac{\partial x}{\partial \xi}\right) \cdot \vec{n}_{i+\frac{1}{2},j} + A_{i-\frac{1}{2},j}\left(\frac{p_{i,j}-p_{i-1,j}}{\Delta d}\frac{\partial x}{\partial \xi}\right) \cdot \vec{n}_{i-\frac{1}{2},j} + \\
A_{i,j+\frac{1}{2}}\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta d}\frac{\partial y}{\partial \eta}\right) \cdot \vec{n}_{i,j+\frac{1}{2}} + A_{i,j-\frac{1}{2}}\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta d}\frac{\partial y}{\partial \eta}\right) \cdot \vec{n}_{i,j-\frac{1}{2}} = \\
\vec{u}^*_{i+\frac{1}{2},j} \cdot \vec{n}_{i+\frac{1}{2},j} + \vec{u}^*_{i-\frac{1}{2},j} \cdot \vec{n}_{i-\frac{1}{2},j} + \vec{u}^*_{i,j+\frac{1}{2}} \cdot \vec{n}_{i,j+\frac{1}{2}} + \vec{u}^*_{i,j-\frac{1}{2}} \cdot \vec{n}_{i,j-\frac{1}{2}}
\end{aligned} \tag{A.43}$$

APÊNDICE B SIMULAÇÃO EFICIENTE DE FUMAÇA EM GRIDS CURVILÍNEOS

Resumo da Dissertação em Português

A qualidade da simulação de fluidos depende de dois fatores importantes: a definição precisa de condições de contorno dos objetos no domínio; e discretização fina em regiões de vorticidade no domínio de simulação. Grids estruturados regulares são empregados tradicionalmente em computação gráfica (FOSTER; METAXAS, 1997; STAM, 1999; FEDKIW; STAM; JENSEN, 2001), simplificando o processo de simulação ao mesmo tempo que não representam adequadamente as condições de contorno. Com a finalidade de mitigar essa situação, varias técnicas foram propostas para melhorar a representação das condições de contorno (LOSASSO; GIBOU; FEDKIW, 2004; ROBLE; ZAFAR; FALT, 2005; BATTY; BERTAILS; BRIDSON, 2007). Enquanto essas técnicas melhoram significativamente os resultados antes obtidos, essas ainda sofrem de limitações inerentes. Grids não estruturados podem representar adequadamente as condições de contorno. No entanto, já que a precisão dos métodos utilizados em grids não estruturados depende de malhas com simplexos isotrópicos, é difícil condensar células em regiões de alta vorticidade para capturar detalhes precisos (WYMAN, 2001).

Nessa tese, nós apresentamos uma abordagem eficiente para criar animações de grande qualidade baseando-se no conceito de grids curvilíneos - uma discretização espacial comumente utilizada na Dinâmica de Fluidos Computacional (DFC). Grids curvilíneos também são conhecidos como grids não regulares estruturados e adaptam-se as formas dos objetos na cena definindo condições de contorno precisamente. Ao contrário de grids não estruturados, grids curvilíneos tem uma topologia bem definida. Essa característica faz que a solução dos sistemas lineares presentes na processo seja simplificada e o custo da resolução das equações seja quase idêntico aos grids regulares tradicionais. Além disso, as células podem ser facilmente condensadas em regiões de alta vorticidade.

Apesar da literatura da DFC apresentar várias técnicas para resolver as equações de Navier-Stokes em grids curvilíneos (CHESSHIRE; HENSHAW, 1990; HENSHAW, 2005b), nenhuma é capaz de ser prontamente utilizada em ambientes de computação gráfica. O nosso método proposto se adapta aos ambientes de computação gráfica e combina em uma única solução eficiente as melhores características de métodos encontrados na DFC: formulação simples, avaliação eficiente, estabilidade incondicional, representação precisa das condições de contorno, suporte ao arranjo descentralizado com coordenadas Cartesianas e suporte a grids sobrepostos.

Para aplicações em computação gráfica, as simulações de fluidos necessitam ser rápidas e estáveis. Para suprir esses requerimentos, nós introduzimos um método Semi-Lagrangiano de advecção incondicionalmente estável baseado em uma transformação de domínio. Além disso, o método proposto é capaz de adotar o arranjo Cartesiano descentralizado através de um novo método da projeção que é simples, robusto e garante a conservação de massa, produzindo campos de velocidades que são livres de oscilações.

A técnica proposta tem custo linear no numero de células do grid e é fácil de implementar. O pipeline é similar ao algoritmo padrão utilizado em grids regulares (STAM, 1999). O passo da projeção requer apenas algumas multiplicações de matrizes adicionais para a transformação de domínio; o passo da projeção requer apenas um ajuste nos coeficientes da matriz de Poisson; e o passo da projeção utiliza interpolações adicionais no campo de pressão.

As contribuições dessa dissertação incluem:

- Uma nova técnica para conservação de massa em grids curvilíneos, utilizando o arranjo descentralizado de variáveis. Nossa solução é mais rápida que o método tradicional utilizado na DFC, baseado no método de D'Yakunov;

- Um algoritmo rápido e incondicionalmente estável para a advecção em grids curvilíneos;

- Uma abordagem eficiente para criar animações de alta qualidade em grids curvilíneos. Nossa técnica possibilita a definição precisa de condições de contorno, obtendo resultados mais precisos do que as técnicas tradicionais baseadas em grids regulares. É também significativamente mais rápida que as técnicas baseadas em grids não estruturados, enquanto produz resultados de qualidade similar.

B.1 Trabalhos relacionados

A simulação de fluidos consiste na avaliação das equações de Navier-Stokes. Existem duas abordagens diferentes para a discretização do espaço: abordagens *Lagrangiana* e *Euleriana*. Métodos Lagrangianos avaliam cada parcela do fluido separadamente, representando todo o fluido como um sistema de partículas. Métodos Eulerianos utilizam grids de pontos fixos distribuídos no domínio de simulação para avaliar as propriedades do fluido. Métodos baseados em sistemas de partículas podem simplificar o processo de simulação, porém os grids são mais eficientes e garantem uma melhor conservação de massa.

A configuração usual de grids consiste em uma subdivisão regular do espaço, discretizando o ambiente de maneira *voxelizada*. Essa abordagem produz representações grosseiras para a geometria do objeto, como mostrado na Figura B.1a. As definições incorretas de geometria introduzem artefatos nas simulações/animações que não desaparecem nem com o refinamento do grid (FELDMAN; O'BRIEN; KLINGNER, 2005; BATTY; BERTAILS; BRIDSON, 2007; ELCOTT et al., 2007; WENDT et al., 2007). Vários trabalhos (FOSTER; FEDKIW, 2001; HOUSTON; BOND; WIEBE, 2003; RASMUSSEN et al., 2004; ROBLE; ZAFAR; FALT, 2005; BATTY; BERTAILS; BRIDSON, 2007) tentar minimizar esses artefatos em grids regulares utilizando informações adicionais como as normais dos objetos, porém eles não são capazes de produzir resultados fisicamente corretos.

Grids não regulares consistem em malhas que contornam precisamente as geometrias dos objetos (KIM; CHOI, 2000; FELDMAN; O'BRIEN; KLINGNER, 2005; KLINGNER

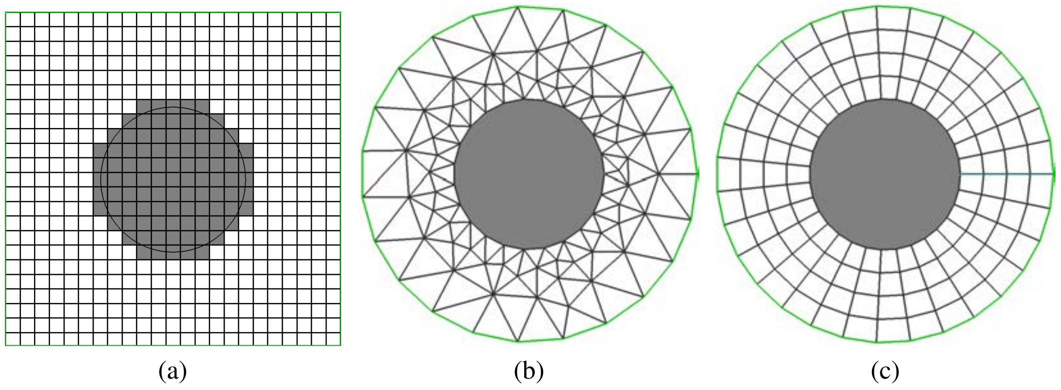


Figura B.1: Técnicas de discretização do espaço: (a) grid regular; (b) grid irregular não estruturado; and (c) grid irregular estruturado.

et al., 2006; TILCH et al., 2008). Essa é uma abordagem comum na DFC, devido a sua precisão. Os grids não regulares podem ser classificados em não-estruturados e estruturados. Os grids não-estruturados discretizam o domínio utilizando triângulos (2-D) ou tetraedros (3-D) (Figura B.1b). Esse método foi introduzido na computação gráfica por Feldman et al. (FELDMAN; O'BRIEN; KLINGNER, 2005) e foi amplamente utilizado e estendido ao longo dos anos (FELDMAN et al., 2005; KLINGNER et al., 2006; CHENTANEZ et al., 2007).

A grande vantagem dos grids não estruturados é que a geração da malha é automática e capaz de gerar malhas para geometrias complexas, com ângulos agudos e concavidades. Como esses grids não têm uma estrutura definida, os algoritmos de solução de sistemas lineares devem ser mais sofisticados e robustos, tornando o método ineficiente em comparação com grids estruturados.

Grids estruturados não-regulares, também conhecidos como grids curvilíneos, são baseados em subdivisões regulares de um espaço Euclidiano que adaptam-se às geometrias dos objetos, preenchendo o domínio sem deixar fendas (Figura B.1c). A sua estrutura regular faz com que o custo seja muito similar ao custo de simulação dos grids regulares tradicionais.

Nós exploramos as propriedades desejáveis dos grids curvilíneos para criar uma abordagem eficiente para simular fluidos. Para isso, nós desenvolvemos uma nova técnica de projeção que funciona em grids descentralizados Cartesianos. Nossa abordagem também utiliza-se de uma transformação de domínio para o passo da advecção. Para dar suporte a cenas dinâmicas, a sobreposição de grids que representam os objetos e o grid que representa o domínio é utilizada. Essa abordagem simplifica ainda mais o processo de geração automática de grids estruturados, já que apenas uma pequena faixa de células ao redor dos objetos é necessária.

B.2 Navier-Stokes em Grids Curvilíneos

A forma diferencial, invíscida e incompressível das equações de Navier-Stokes é escrita como

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{f} \tag{B.1}$$

e

$$\nabla \cdot \vec{u} = 0, \tag{B.2}$$

onde $\vec{u}$ e p são os campos de velocidade e pressão, respectivamente, ρ é a densidade do fluido, e $\vec{f}$ representa as forças de corpo que atuam no fluido. As Equações (B.1) e (B.2) são conhecidas como as equações de *conservação de momentum* e *conservação de massa*. A forma tradicional de resolver ambas as equações (B.1) e (B.2) é através do método da projeção (CHORIN, 1968), no qual consiste em três fases: (i) advecção, (ii) resolução da pressão, e (iii) projeção da velocidade. Stam [1999] introduziu esse método a computação gráfica, e apresenta uma descrição completa do algoritmo.

B.2.1 Advecção Semi-Lagrangiana em grids Curvilíneos

Nessa tese, estendemos a abordagem de Karpik et al. (KARPIK; CROCKETT, 1997) para atualizar dinamicamente o campo de velocidades utilizando uma técnica de transformação de domínio. Essa técnica baseia-se em transformar o grid não-regular (domínio físico) para um grid canônico regular (domínio computacional). Com isso, a avaliação do algoritmo Semi-Lagrangiano é feita no grid canônico utilizando-se a formulação tradicional. A equação que expressa a transformação de velocidades do domínio físico para o domínio computacional é dada por

$$\vec{u}_{\xi\eta\tau} = T_{cp}^{-1}\vec{u}_{ijk}, \tag{B.3}$$

onde $\vec{u}_{ijk}$ é a velocidade no domínio físico, $\vec{u}_{\xi\eta\tau}$ é a velocidade no domínio computacional. A matriz de transformação T_{cp}^{-1} é

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \mathbf{g_x^{(1)}} & \mathbf{g_x^{(2)}} & \mathbf{g_x^{(3)}} \\ \mathbf{g_y^{(1)}} & \mathbf{g_y^{(2)}} & \mathbf{g_y^{(3)}} \\ \mathbf{g_z^{(1)}} & \mathbf{g_z^{(2)}} & \mathbf{g_z^{(3)}} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, \tag{B.4}$$

sendo que $\mathbf{g_x^{(i)}}, \mathbf{g_y^{(i)}}, \mathbf{g_z^{(i)}}$ são os elementos das bases contravariantes do grid. O Algoritmo 4 sumariza a técnica de advecção em grids curvilíneos.

Algorithm 4 Algoritmo Semi-Lagrangiano para Grids Curvilíneos

```
/* Velocidades Cartesianas (eixos ijk) armazenadas nos centros das faces do grid */
para todas as células do grid no domínio físico faça
    /* Interpolar os componentes não conhecidos da velocidade no centro da face*/
    $\tilde{u}_{ijk} \leftarrow$ interpolaVelocidade($\vec{u}^n_{ijk}, x$)
    /* Transformar a velocidade para o domínio computacional (eixos $\xi\eta\tau$) utilizando Eq. B.3*/
    $\tilde{u}_{\xi\eta\tau} \leftarrow$ transformarParaDominioComputacional($\tilde{u}_{ijk}$)
    /* Calcular a posição $x_p$ no passo de tempo anterior */
    $x_{\xi\eta\tau} \leftarrow$ encontrarPosiçãoAnterior ($x, \tilde{u}_{\xi\eta\tau}, \Delta t$)
    /* Estimar o campo de velocidades $u^{*(n+1)}_{ijk}$ na posição anterior*/
    $u^{*(n+1)}_{ijk}(x) \leftarrow$ interpolaVelocidade($\vec{u}^n_{ijk}, x_{\xi\eta\tau}$)
fim para
```

B.2.2 Resolução da pressão em grids Curvilíneos

Na computação gráfica, a forma tradicional de representar a *conservação de massa* para fluidos incompressíveis é garantindo que o divergente do campo de velocidades seja igual a zero (Eq. (B.2)). No entanto, a sua forma integral

$$\int_V \nabla \cdot \vec{u}\, \mathrm{d}V = \int_S \vec{u} \cdot \vec{n}\, \mathrm{d}S = 0, \tag{B.5}$$

é mais conveniente para nossa solução, já que simplifica a representação dos operadores de divergente e gradiente em sistemas de coordenadas genéricos. Na Eq. (3.12), V e S denotam o volume e a área das células que estão sendo avaliadas, respectivamente. Usando o método da projeção para acoplar a conservação de momentum (Eq. (B.1)) e massa (Eq. (B.5)), tem-se

$$\int_S \vec{u}^{(n+1)} \cdot \vec{n} \, \mathrm{d}S = \int_S \vec{u}^{*(n+1)} \cdot \vec{n} \, \mathrm{d}S - \Delta t \int_S \frac{\partial p^{(n+1)}}{\partial \vec{n}} \, \mathrm{d}S = 0, \tag{B.6}$$

onde $\vec{n}$ representa as normais unitárias das faces avaliadas. Por fim, o campo de pressões que garante a conservação de massa é obtido por

$$\int_S \frac{\partial p^{(n+1)}}{\partial \vec{n}} \, \mathrm{d}S = \frac{\int_S \vec{u}^{*(n+1)} \cdot \vec{n} \, \mathrm{d}S}{\Delta t}. \tag{B.7}$$

B.2.3 Projeção da Velocidade em grids Curvilíneos

Substituindo-se o o campo de pressões obtido com a solução de Eq. (B.7) em Eq. (B.6), obtemos os fluxos que satisfazem a conservação de massa em cada célula. Como armazenamos apenas as componentes Cartesianas das velocidades nos centros de cada face, precisamos transformar os fluxos projetados para atualizar os componentes de velocidade. Shyy and Vu (SHYY; VU, 1991) recuperam os componentes de velocidade através de um algoritmo iterativo baseado no método de D'Yakunov (CONCUS; GOLUB, 1972). Apesar desse método ser preciso, ele é ineficiente, sofrendo restrições para grids com um número grande de células.

Portanto, nós criamos um método novo para projetar as velocidades em grids descentralizados Cartesianos. Nossa solução é mais rápida, simples e mais fácil de implementar que o método de D'Yakunov. *O método proposto consiste em corrigir as velocidades não projetadas avaliando as derivadas das pressões em todas as direções Cartesianas.* Usando a regra da cadeia, podemos escrever

$$\begin{aligned}
\vec{u}_x &= \vec{u}_x^* - \Delta t \frac{\partial p}{\partial x} = \vec{u}_x^* - \Delta t \left(\frac{\partial p}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \tilde{p}}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial \tilde{p}}{\partial \tau}\frac{\partial \tau}{\partial x}\right)\mathbf{i}, \\
\vec{u}_y &= \vec{u}_y^* - \Delta t \frac{\partial p}{\partial y} = \vec{u}_y^* - \Delta t \left(\frac{\partial p}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial \tilde{p}}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial \tilde{p}}{\partial \tau}\frac{\partial \tau}{\partial y}\right)\mathbf{j}, \\
\vec{u}_z &= \vec{u}_z^* - \Delta t \frac{\partial p}{\partial z} = \vec{u}_z^* - \Delta t \left(\frac{\partial p}{\partial \tau}\frac{\partial \tau}{\partial z} + \frac{\partial \tilde{p}}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial \tilde{p}}{\partial \eta}\frac{\partial \eta}{\partial z}\right)\mathbf{k},
\end{aligned} \tag{B.8}$$

onde (ξ, η, τ) são as linhas do grid no domínio físico. A versão 2-D dessas equações apenas omite o terceiro termo (τ) dentro dos parênteses. A Figura B.2 mostra a atualização de cada componente Cartesiana para uma célula no grid curvilíneo.

B.3 Decomposição de domínios

Para suportar de maneira eficiente cenas com múltiplos objetos, nós decompomos o domínio de simulação utilizando a técnica de *grids sobrepostos* (BRANDT, 1977; CHESSHIRE; HENSHAW, 1990). Cada objeto é representado separadamente por um grid curvilíneo que discretiza-o precisamente. Esses grids são sobrepostos em um grid regular de fundo, que delimita o domínio de simulação. A Figura B.3 ilustra conceitualmente a técnica de decomposição de domínios.

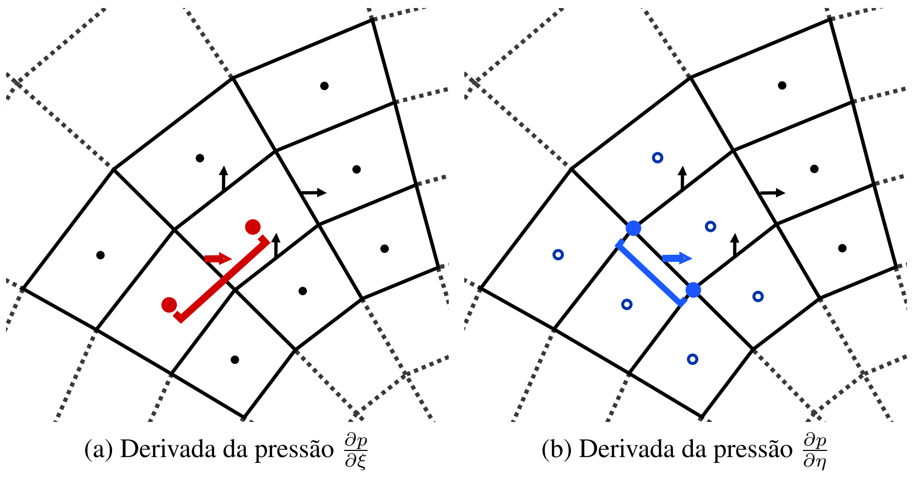

(a) Derivada da pressão $\frac{\partial p}{\partial \xi}$ (b) Derivada da pressão $\frac{\partial p}{\partial \eta}$

Figura B.2: Atualizando o componente x da velocidade Cartesiana na face l. (a) As derivadas das pressões são avaliadas com os valores da pressão no centro das células que compartilham a face l (pontos vermelhos). (b) As derivadas das pressões são avaliadas com os valores da pressão armazenados no centro das células adjacentes (pontos azuis).

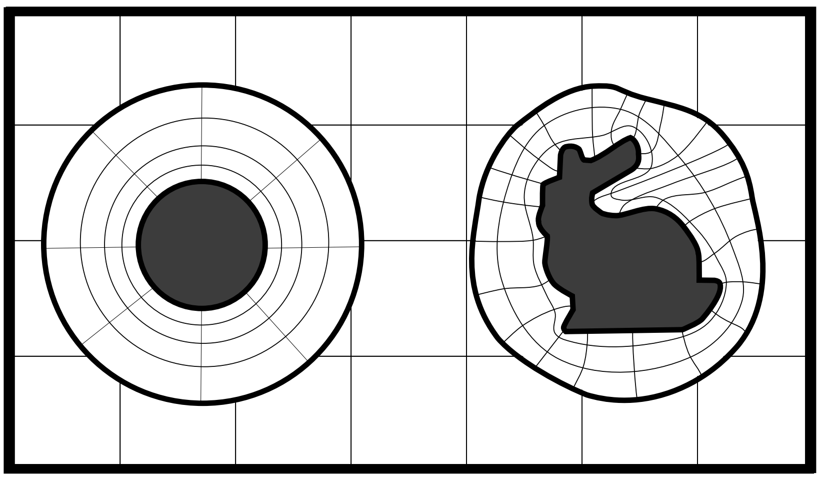


Figura B.3: Configuração da técnica de grids sobrepostos. Um grid regular discretiza o domínio de simulação e cada objeto é representado por um grid curvilíneo.

Inicialmente, a advecção é realizada no grid de fundo; as velocidades intermediárias obtidas são interpoladas nas bordas dos grids curvilíneos. O passo de advecção é realizado nos grids curvilíneos; o resultado é interpolado de volta para o grid regular de fundo. O passo de resolução da pressão é similar, porém empregamos um algoritmo de multigrid para resolução do sistema linear. O passo da projeção é feito separadamente para todos os grids.

B.4 Resultados

A Figura B.4 (esquerda) mostra uma simulação 2-D em um canal com múltiplos obstáculos. Esse exemplo ilustra a qualidade dos nossos resultados e a flexibilidade da técnica de discretizar domínios compostos. A Figura B.4 (direita) mostra uma simulação 3-D de fumaça em um túnel de vento, para inspeção visual das propriedades aerodinâmicas de um carro.

A Figura B.5 demonstra a capacidade da nossa abordagem de representar as condições de contorno de forma correta. Nesse exemplo, comparamos os resultados obtidos com os grids regulares (direita) e grids curvilíneos (direita) para uma seção transversal de uma asa. Note que no primeiro caso, as condições de contorno estão incorretas, gerando uma região de alta turbulência na parte posterior da asa. Nos nossos resultados, a fumaça apresenta um comportamento bem definido, sem turbulências aparentes.

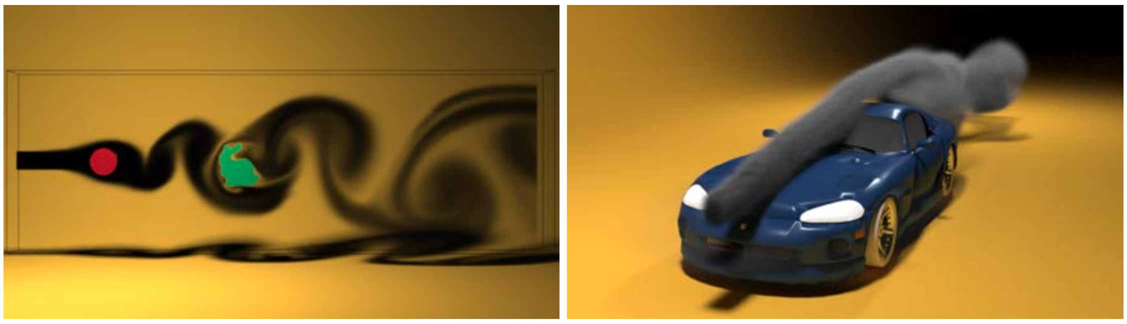


Figura B.4: Exemplos de simulação de fumaça produzidos com a nossa técnica . (esquerda) Simulação 2-D em um canal com múltiplos obstáculos. (direita) Simulação 3-D de um um túnel de vento visualizar as propriedades aerodinâmicas de um carro.

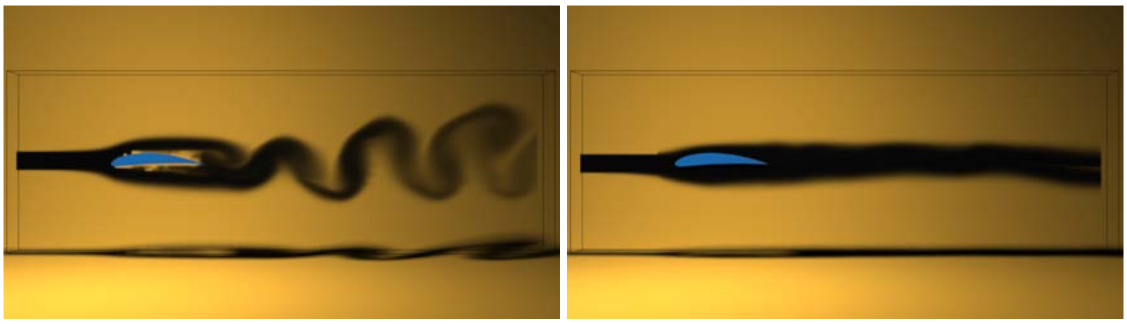


Figura B.5: Simulação de um escoamento utilizando uma seção transversal de uma asa: grid regular (esquerda); grid curvilíneo (direita).