Pacific Graphics 2013
B. Levy, X. Tong, and KK. Yin
(Guest Editors)

Volume 32 (2013), Number 7

Efficient Smoke Simulation on Curvilinear Grids

Vinicius C. Azevedo† and Manuel M. Oliveira‡

Instituto de Informática – UFRGS

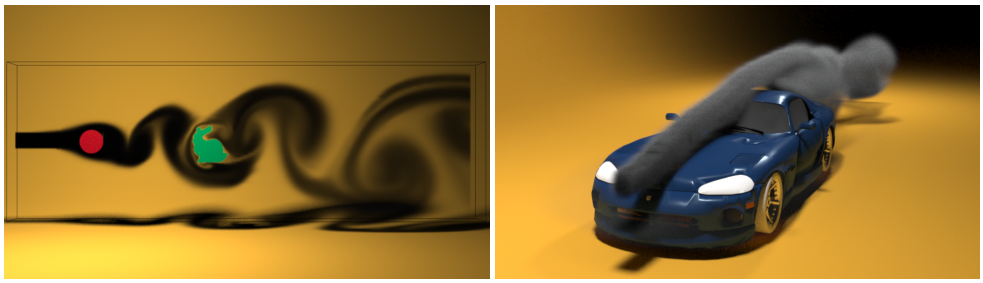

Figure 1: Examples of smoke simulations produced with our technique. (left) 2-D simulation on a channel with multiple obstacles. (right) 3-D smoke simulation in a wind tunnel to visually evaluate a car's aerodynamics properties.

Abstract

We present an efficient approach for performing smoke simulation on curvilinear grids. Our technique is based on a fast unconditionally-stable advection algorithm and on a new and efficient solution to enforce mass conservation. It uses a staggered-grid variable arrangement, and has linear cost on the number of grid cells. Our method naturally integrates itself with overlapping-grid techniques, lending to an efficient way of producing highly-realistic animations of dynamic scenes. Compared to approaches based on regular grids traditionally used in computer graphics, our method allows for better representation of boundary conditions, with just a small increment in computational cost. Thus, it can be used to evaluate aerodynamic properties, possibly enabling unexplored applications in computer graphics, such as interactive computation of lifting forces on complex objects. We demonstrate the effectiveness of our approach, both in 2-D and 3-D, through a variety of high-quality smoke animations.

1 Introduction

The quality of 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 [Sta99, FSJ01] simplifies the simulation process at the expense of using poorly-defined boundary conditions. To mitigate this situation, several techniques have been proposed to improve the representation of boundary cells [LGF04, RZF05, BBB07a]. While they significantly improve previous results, they have their own limitations (Section 2). Unstructured grids, on the other hand, can precisely represent boundaries. However, since unstructured grids have no organized structure pattern, more sophisticated and robust algorithms are required to solve the resulting system of equations, causing its solution to be slower than for structured grids.

We present an efficient approach for creating high-quality flow simulations based on *curvilinear grids* – a space discretization commonly used in Computational Fluid Dynamics (CFD) [Sor02]. 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,

† e-mail: vcazevedo@inf.ufrgs.br
‡ e-mail: oliveira@inf.ufrgs.br

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, where increased turbulence and high-vorticity tend to appear.

Although the CFD literature presents several methods for solving the Navier-Stokes equations on curvilinear grids [SDB83, HS06], none of them is readily applicable to computer graphics. *Our method adapts and combines in a single efficient solution their best features*: simple formulation, efficient evaluation, unconditional stability, accurate representation of boundary conditions, and support to staggered-grid variable arrangement and overlapping grids.

For graphics applications, fluid simulation needs to be both stable and fast. To meet these requirements, *we introduce a fast unconditionally-stable semi-Lagrangian method based on a domain transformation* (from the physical grid to a canonical computational one). We are also able to maintain a Cartesian staggered arrangement by introducing *a simple and robust mass-conservation method that produces oscillation-free velocity fields*, which is a key contribution of our work.

Figure 1 presents two examples of flow simulations created with our approach. On the left, a 2-D simulation on a channel with multiple obstacles shows that it properly maintains boundary conditions. On the right, we see a 3-D smoke simulation on a wind tunnel. These examples illustrate the potential and flexibility of our method. The use of overlapping grids lends to an efficient way of producing highly-realistic animations of dynamic scenes. Compared to previous techniques used in computer graphics, ours allows one to better explore scenarios where physical accuracy is necessary, such as real-time calculation of lifting forces on aerodynamic objects.

The **contributions** of our work include:

- A new technique to enforce mass conservation on curvilinear grids with a staggered variable arrangement (Section 3.3). Our solution is faster than the traditional curvilinear one used in CFD, based on D'Yakunov's method [CG72];
- A fast unconditionally-stable advection algorithm for curvilinear grids (Section 3.1);
- An efficient approach for creating high-quality flow simulations on curvilinear grids (Sections 3.1 to 3.3). Our technique allows for precise definition of boundary conditions, leading to more realistic simulations and animations than existing regular-grid-based approaches. It is also faster than unstructured-grid-based techniques, while producing results of similar quality.

2 Related Work

Fluid simulation consists of evaluating the Navier-Stokes equations, which is basically done using one of two popular approaches: Lagrangian, or Eulerian. *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. Particle methods can simplify the simulation process, but grids are more efficient and better enforce incompressibility [Bri08].

The usual grid layout consists of a regular subdivision, discretizing the environment in a voxelized fashion. Although common and reliable, it produces rough representations for object geometry, as shown in Figure 2a. The resulting incorrect boundaries introduce noticeable artifacts in the simulations/animations, which do not vanish with increased grid resolution [FOK05, BBB07a]. Foster and Fedkiw [FF01] tried to minimize these artifacts using the obstacle's normal vectors to enforce appropriate velocity boundary conditions. This idea was further extended by Houston et al. [HBW03] and by Rasmussen et al. [REN*04]. However, the voxelized pressure solver still produces objectionable artifacts unless high-resolution grids are used [BBB07a].

Adaptive refinement techniques [LGF04, LZF10] try to optimize the domain cell distribution concentrating them near object boundaries. While octrees [LGF04] can improve the use of computational resources, it leads to non-symmetric systems in the pressure equation. To overcome this, the authors simplify the representation of the pressure gradient. This drops the projection step to first order accuracy in space, hindering the overall exactness of the method. Moreover, object boundaries are still represented in a voxelized fashion, affecting boundary conditions.

Immersed-boundary (IB) methods use clever observations to avoid voxelized pressure solvers [RZF05, BBB07a]. Roble et al. [RZF05] modified the regular-grid finite-difference method, allowing voxels to "deform" and align to boundaries, producing a divergence-free solution. Batty et al. [BBB07a] proposed a variational approach to handle irregular boundaries in regular grids. However, since those methods modify the pressure equation on boundaries with a minimization function, they fail to reproduce physically accurate aerodynamic effects, as shown in Section 5.

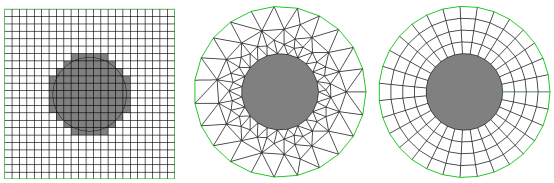

Figure 2: Space-discretization techniques: (a) regular grid; (b) unstructured grid; and (c) non-regular structured grid.

Unstructured grids often discretize the solution domain using triangles (2-D) or tetrahedra (3-D) (Figure 2b). They were introduced to computer graphics by Feldman et

al. [FOK05]. 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 [FOKG05, KFCO06, CFL*07] using a mesh re-generation technique, which can take up to forty percent of the total simulation time [KFCO06]. Unstructured grids were further combined to Immersed Boundary methods [BXH10], to enable spatially adaptive liquid simulation with more accurate enforcement of air and liquid interfaces.

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 [Wym01].

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 2c). They are constructed using quadrilaterals (in 2-D) or hexahedra (in 3-D). Their regular structure makes the cost of the flow solver nearly identical to the ones used with regular grids.

Stam [Sta03] used Catmull-Clark surfaces to simulate flows on surfaces of arbitrary topology. Unfortunately, Stam's approach cannot generate arbitrary curvilinear grids to precisely represent boundary conditions, and does not support local cell refinements. According to our experience, the use of this method on general curvilinear grids introduces velocity oscillations in the simulations. Moreover, since the method does not use a domain transformation in the advection phase, a local search to find the correct cell index of the backtracked particle is needed, negatively affecting its performance. It is not clear how to extend this solution to 3-D. [EQYF13]

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 (MAC) staggered-Cartesian variable arrangement (Section 3.2). Our approach also uses a fast unconditionally-stable advection algorithm for curvilinear grids (Section 3.1). To handle dynamic scenes, we overlap (small) curvilinear grids representing objects' boundaries on top of a regular grid that defines the simulation domain (Section 4).

3 Navier-Stokes on Curvilinear Grids

The differential form of the inviscid, incompressible Navier-Stokes equations are written as

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

and

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

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. Equations (1) and (2) are called the *momentum conservation* and the *mass conservation* equations.

The traditional way for simultaneously solving Eqs. (1) and (2) is using the *projection method* [Cho68], 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 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 $p^{(n+1)}$. Stam [Sta99] introduced this method to computer graphics and presents a good description of the algorithm. Sub-sections 3.1 to 3.3 present the details of our projection method for curvilinear grids.

3.1 Semi-Lagrangian Advection on Curvilinear Grids

The *semi-Lagrangian method* [Rob81] is a classic algorithm used in computer graphics for the advection phase in fluid simulation [Sta99]. 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:

$$\vec{u}^{*(n+1)}(x) = \vec{u}^{(n)}(x_p), \tag{3}$$

$$x_p = x - \vec{v}^{(n)}(x)\Delta t, \tag{4}$$

where $\vec{v}^{(n)}(x)$ is q's velocity at position x and time $t^{(n)}$. $\vec{v}^{(n)}(x)$ is interpolated from the values of $\vec{u}^{(n)}$ stored on the Eulerian grid.

While the semi-Lagrangian method is straightforward on regular grids, it is not directly applicable to non-regular ones. 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.

To simplify the advection of densities through a curvilinear grid, CFD researchers [KC97] transform the (*physical-domain*) non-regular grid Ω onto a (*computational-domain*) canonical regular grid Ω' (Figure 3). 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 [Bar84]. Intuitively, they map each cell (a quadrilateral in Figure 3 (left)) from the original mesh to a canonical cell (a unit square in Figure 3 (right)). Note that these transformations are homeomorphisms and vary from cell to cell.

We extend the approach of Karpik and Crockett [KC97] to dynamically update the intermediary velocity field using a domain transformation. Thus, we are able to achieve the same simplicity, stability, and speed of the standard semi-Lagrangian method traditionally used on regular grids.

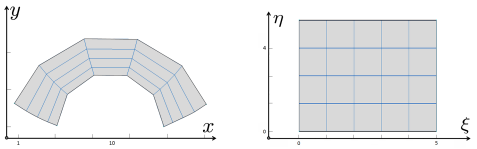

Figure 3: 2-D representation of the physical domain Ω (left) and canonical computational domain Ω' (right).

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

$$\vec{u} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = \vec{u}_{ijk}, \tag{5}$$

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

$$\vec{u} = u_1\mathbf{g^1} + u_2\mathbf{g^2} + u_3\mathbf{g^3} = \vec{u}_{\xi\eta\tau}, \tag{6}$$

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

$$\mathbf{g^1} = (\frac{\partial x}{\partial \xi}, \frac{\partial y}{\partial \xi}, \frac{\partial z}{\partial \xi}),\ \mathbf{g^2} = (\frac{\partial x}{\partial \eta}, \frac{\partial y}{\partial \eta}, \frac{\partial z}{\partial \eta}),\ \mathbf{g^3} = (\frac{\partial x}{\partial \tau}, \frac{\partial y}{\partial \tau}, \frac{\partial z}{\partial \tau}). \tag{7}$$

From $\mathbf{g^i}$ one can compute a unit basis as

$$\mathbf{g^{(1)}} = \frac{1}{h_1}\mathbf{g^1},\quad \mathbf{g^{(2)}} = \frac{1}{h_2}\mathbf{g^2},\quad \mathbf{g^{(3)}} = \frac{1}{h_3}\mathbf{g^3}, \tag{8}$$

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

$$h_i = ((\mathbf{g^i})^T \cdot \mathbf{g^i})^{\frac{1}{2}}. \tag{9}$$

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

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \mathbf{g_x^{(1)}} & \mathbf{g_x^{(2)}} & \mathbf{g_x^{(3)}} \\ \mathbf{g_y^{(1)}} & \mathbf{g_y^{(2)}} & \mathbf{g_y^{(3)}} \\ \mathbf{g_z^{(1)}} & \mathbf{g_z^{(2)}} & \mathbf{g_z^{(3)}} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, \tag{10}$$

and it is valid only if the 3x3 matrix (T_{cp}) shown in Eq. (10) is non-singular, *i.e.*, the vectors $\{\mathbf{g^i}\}$ 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. (10):

$$\vec{u}_{\xi\eta\tau} = T_{cp}^{-1}\vec{u}_{ijk}. \tag{11}$$

The Cartesian components of $\vec{u}_{ij}$ are stored at the center of the cell faces of the (canonical) computational grid in a Cartesian staggered fashion. To perform the semi-Lagrangian step in the computational grid, for each cell face, one evaluates Eq. (11) to obtain $\vec{u}_{\xi\eta\tau}$ (from the components of $\vec{u}_{ij}$ stored on the adjacent cells sharing the face - Figure 4 (right)). Then, one evaluates Eq. (4) with $\vec{v}^n = \vec{u}^n_{\xi\eta}$ to obtain the position $x_{\xi\eta\tau}$. The estimated velocity at $x_{\xi\eta\tau}$ is then used as the new intermediate velocity $\vec{u}_{ij}^{*(n+1)}$ (Eq. (3)). Note that $\vec{u}_{\xi\eta\tau}$ *is only used in the back-trajectory step*; the interpolation at $x_{\xi\eta\tau}$ uses the stored velocities defined in Cartesian coordinates. This allows us to directly adapt higher-order methods, such as the modified MacCormack [SFK*08].

Eq. (7) expresses the contravariant basis which will be used to transform the velocity components. This basis is defined in the cell interior and undefined at cell boundaries. At the boundaries, we always use the basis defined for the next cell (i-th cell + 1). For example, for a face shared by a cell on its left and by a cell on its right, the contravariant basis defined for the cell on its right is used; for a face shared by a cell below it and by a cell above it, the basis defined for the top cell is used. The situation for cells in front and behind a face is similar.

3.2 Pressure Solving on Curvilinear Grids

The integral form of the mass conservation equation (Eq. (12)) is more convenient for our solution, as it simplifies the representation of divergence and gradient operators on general coordinate systems.

$$\int_V \nabla \cdot \vec{u}\, dV = \int_S \vec{u} \cdot \vec{n}\, dS = 0. \tag{12}$$

In Eq. (12), V and S denote the volume and the faces of the cell under evaluation, respectively. Using the projection method to couple momentum (Eq. (1)) and mass conservation (Eq. (12)) yields

$$\int_S \vec{u}^{(n+1)} \cdot \vec{n}\, dS = \int_S \vec{u}^{*(n+1)} \cdot \vec{n}\, dS - \Delta t \int_S \frac{\partial p^{(n+1)}}{\partial \vec{n}}\, dS = 0, \tag{13}$$

where $\vec{n}$ represents the faces' corresponding unit normal vectors. The pressure field p that satisfies Eq. (13) can be expressed as

$$\int_S \frac{\partial p^{(n+1)}}{\partial \vec{n}}\, dS = \frac{\int_S \vec{u}^{*(n+1)} \cdot \vec{n}\, dS}{\Delta t}. \tag{14}$$

The left-hand side of Eq. (14) can be discretized as

$$\int_S \frac{\partial p^{(n+1)}}{\partial \vec{n}}\, \mathrm{d}S \approx \left[(p_c^{(n+1)} - p_{adj}^{(n+1)}) \frac{(\vec{\xi}_c \cdot \vec{n}_l)}{||\vec{\xi}_c||} \right] A_l, \quad (15)$$

where p_c is the pressure evaluated at the current cell C_c, and p_{adj} is the pressure evaluated at the adjacent cell C_{adj} that shares face l with C_c. A_l is l's area, and $\vec{\xi}_c$ is the vector from the center of C_c to the center of C_{adj} (dashed vector in Figure 4 (left)). Finally, we discretize the right-hand side

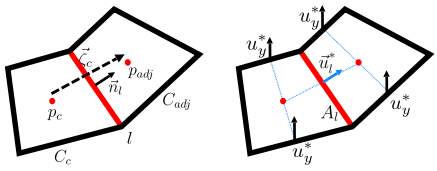

Figure 4: (left) Elements required to discretize the pressure derivative along a cell-face normal. (right) Interpolation of the vertical velocity component at the center of a cell face. Due to the staggered velocity scheme, the vertical velocity is interpolated from nearby locations (u_y^*) producing the full velocity at face l, $\vec{u}_l^* = (u_x, \tilde{u}_y)$.

of Eq. (14) as

$$\int_S \vec{u}^{*(n+1)} \cdot \vec{n}\ \mathrm{d}S \approx \sum_{l \in \text{faces}} (\vec{u}_l^{*(n+1)} \cdot \vec{n}_l) A_l, \quad (16)$$

where the intermediate velocity $\vec{u}_l^{*(n+1)}$ is evaluated at the center of the cell face l. The unkown components of $\vec{u}_l^{*(n+1)}$ are interpolated from neighbor cells, as done for regular grids (Figure 4 (right)).

3.3 Velocity Projection on Curvilinear Grids

When substituting in Eq. (13) the pressure field obtained with the solution of Eq. (14), one obtains the fluxes (across the cells' faces) that **satisfy the incompressibility constraint**. Since the grid stores the Cartesian components of the velocity, one would need to transform the projected fluxes back into velocity components. Shyy and Vu [SV91] recover the velocity components using an iterative scheme based on D'Yakunov's method [CG72]. Although accurate, such method tends to be slow, since it requires several iterations on each time-step of the projection phase, 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:

$$\begin{aligned}
\vec{u}_x &= \vec{u}_x^* - \Delta t \frac{\partial p}{\partial x} = \vec{u}_x^* - \Delta t \left(\frac{\partial p}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \tilde{p}}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial \tilde{p}}{\partial \tau}\frac{\partial \tau}{\partial x}\right)\mathbf{i}, \\
\vec{u}_y &= \vec{u}_y^* - \Delta t \frac{\partial p}{\partial y} = \vec{u}_y^* - \Delta t \left(\frac{\partial p}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial \tilde{p}}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial \tilde{p}}{\partial \tau}\frac{\partial \tau}{\partial y}\right)\mathbf{j}, \\
\vec{u}_z &= \vec{u}_z^* - \Delta t \frac{\partial p}{\partial z} = \vec{u}_z^* - \Delta t \left(\frac{\partial p}{\partial \tau}\frac{\partial \tau}{\partial z} + \frac{\partial \tilde{p}}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial \tilde{p}}{\partial \eta}\frac{\partial \eta}{\partial z}\right)\mathbf{k},
\end{aligned} \quad (17)$$

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). We explain how to update the velocity component along the x direction in 2-D (highlighted arrows in Fig. 5); 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 Figure 5 (left)). 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 5 (right)). Each such value is then interpolated from the pressure values stored at the centers of all cells sharing the corresponding vertex (small green circles in Figure 5 (right)). The *tilde* in $\frac{\partial \tilde{p}}{\partial \eta}$ indicates that we use interpolated pressure values on face l. Thus, we compensate the lack "de-interpolation" of the Cartesian velocity components traditionally used in CFD with a few 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 orthogonal grids, $\frac{\partial \xi}{\partial x} + \frac{\partial \eta}{\partial x} + \frac{\partial \tau}{\partial x} = 1$ (likewise for y and z, as well as for 2-D grids). The first property allows our method to be used on regular grids without any adaptation. Moreover, it can be used on orthogonal grids with variable cell sizes, which is useful for concentrating cells in regions of interest of the simulation domain. The second property shows that the contribution of the pressure correction will be physically consistent. All simulations (both 2-D and 3-D) shown in the paper and accompanying video were produced using our velocity projection technique.

4 Domain Decomposition

To efficiently support multiple objects with independent and dynamic rigid-body motions, we decompose the simulation domain using *overlapping or Chimera grids* [SDB83, WP00]. Each object is represented by a curvilinear grid that tightly fits its boundary. Such grids are then superimposed on a regular one that delimits the simulation domain. A representation of our overlapping-grid setup is shown in Figure 6.

Some methods [DMYN08, EQYF13] use regular grids for both background and superimposed obstacles. Dobashi et al. cut grids holes on the background grid, using boundary cells as transfer mechanism through grids. English et

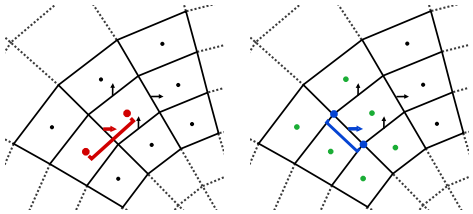

Figure 5: Updating the x-component of the Cartesian velocity (highlighted arrows) at a given cell face l. (left) $\frac{\partial p}{\partial \xi}$ pressure derivatives evaluated from values stored at the centers of the two cells sharing l (red dots). (right) $\frac{\partial \tilde{p}}{\partial \eta}$ pressure derivatives evaluated from interpolated values (solid blue dots) from the pressure values stored at the centers of the surrounding cells (green circles).

al. [EQYF13] uses grid patching to join the overlapped grids into a single mesh. Our method is similar to [DMYN08], but we use a more accurate approach for exchanging flow information among the grids (Figure 6 (right)), which we adapted from [Hen05].

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)) 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. A detailed description of our multigrid approach is provided in the supplemental materials. After the pressure-solving step, the velocity fields are independently projected in each grid.

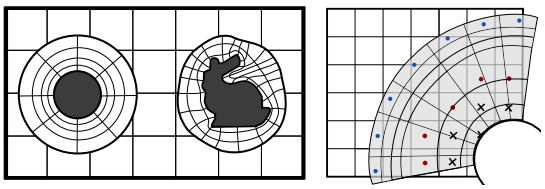

Figure 6: Overlapping grids. (left) A regular grid discretizes the simulation domain and each object has tightly-fit curvilinear grid. (right) Interpolation cells are used for exchanging flow information among grids and for setting boundary conditions. Curvilinear-grid interpolation cells (●); regular-grid interpolation cells (●); unused regular-grid cells (×).

The use of overlapping grids requires the simulation to transfer flow information among 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 6 (right)). *Interpolation cells* are the ones at the edges of the overlapping

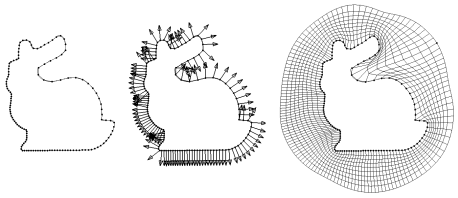

Figure 7: Automatic generation of periodic curvilinear grid by extruding the mesh along the its normal vectors. Grid created using the software Gridgen by Pointwise, Inc.

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 use Dirichlet boundary conditions).

Given an overlapping/interpolation band, its outer edge consists of curvilinear-grid interpolation cells, while the inner edge is made of regular-grid interpolation cells. Figure 6 (right) 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 × in Figure 6 (right). All remaining cells are discretization ones. These and interpolation cells are used for simulation.

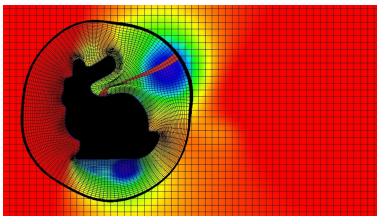

Figure 8: Pressure fields are continuous across grids. Dark red cells along the Northeast direction in the bunny grid highlight the borders of this periodic grid.

Overlapping bands provide the mechanism for proper information exchange, ensuring continuous velocity and pressure fields across grids, as illustrated in Figure 8. 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 strategies 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.

5 Results

We have implemented the techniques described in the paper 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 [BG12]. The CPU version uses multigrid and currently is available only for 2-D. The 2-D curvilinear grids used for the examples in the paper and accompanying materials were created automatically using the software Gridgen by Pointwise Inc [Poi12]; For 3D grids, we use a 3D Studio Max® plugin to automatically generate a pair of NURBS surfaces that tightly wraps the target 3-D geometry (see supplemental video). We then use Gridgen to create an optimized 3-D grid from these NURBS.

Figure 1 (left) shows a 2-D simulation on a channel with multiple obstacles. It illustrates the quality of our results and the flexibility of our technique. The curvilinear grids for the circle and the bunny overlap a regular grid that delimits the simulation domain.

Figure 12 compares our technique to the variational approach [BBB07a] and to the regular grid method [Sta99] for a variety of 2-D examples: airfoil (NACA 6412), a circle, and a cross section of the Stanford bunny. For each example, the images show all methods at the same instant of the simulation. For the variational approach, we use the sample code available in the authors' project website [BBB07b]. All simulations were performed using a modified MacCormack method [SFK*08]. The variational and regular-grid methods use a 248×82 grid (total of 20,336 cells) for all examples. To maintain the total number of cells approximately the same, our overlapping-grid approach uses a 193×65 regular grid (12,545 cells) to define the simulation domain. The curvilinear grids of the airfoil ($192 \times 33 = 6,336$ cells), sphere ($192 \times 33 = 6,336$ cells), and bunny ($128 \times 33 = 4,224$ cells) are then superimposed on it. Thus, our corresponding simulations use a total of 18,881, 18,881, and 16,769 cells, respectively.

The first column of Figure 12 shows the airfoil oriented at zero degrees. Due to its aerodynamic properties, one should expect little or no turbulence in the passing flow, which can be observed in the result produced by our technique. The poorly-enforced boundary conditions of the regular grid causes the flow not to follow the airfoil's boundaries and introduces high-vorticity regions behind it. The variational approach improves this situation, but does not completely eliminates these problems.

The second column of Figure 12 shows a similar comparison for the airfoil oriented at 10 degrees. The regular grid lends to an incorrect simulation. The variational approach improves the results, but does not fix the problem. The accompanying video shows that in our simulation the fluid is tightly coupled to the airfoil, and flows over it at higher speed than under it, a condition required to produce lifting.

The last two columns of Figure 12 compare the three methods for general object boundaries. The curvilinear discretization of the bunny and sphere lends to better simulation results. Our approach is able to generate correct density fields despite the fact that the shapes of the individual cells in the bunny's grid are highly irregular (see Figure 8).

Figure 1 (right) shows a 3-D smoke simulation in a wind tunnel for visual inspection of a car's aerodynamics properties. Figure 10 shows a side view of this simulation at a different time step. Note the 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 11. 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 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.

Figure 9 shows frames of an animation of two moving objects (circle and bunny). The smoke is nicely displaced as the objects move. We exploit the use of overlapping grids to efficiently handle dynamic environments. For this, we use an approach similar to the one described in [DMYN08].

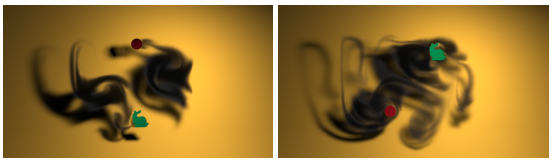

Figure 9: Frames of an animation showing two moving objects (circle and bunny). We exploit the use of overlapping grids to efficiently handle dynamic scenes.

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 i7-2600 3.40 GHZ CPU (8 GB of RAM) and a GeForce GTX 460 (1 GB of RAM). For the 2-D examples, the regular-grid approach uses a multigrid implementation running on the CPU. Column "*Press + Proj*" combines the execution times for the pressure solving and velocity projection phases.

The numbers in Table 1 show that, normalized by the number of grid cells, the performance of our technique is similar to the one of the regular-grid approach. The main benefit of our technique is that it lends to more accurate simulation results. For instance, the 2-D simulation of a circular obstacle (second row of Table 1, third column of Figure 12) uses a total number of 20,667 cells for the regular grid, and 19,206 cells for the overlapping configuration used by our method. For this example, each step of the simulation takes 0.0481 seconds for the regular grid and 0.0491 seconds for our method. Our mass conservation solution is quite efficient, requiring only 0.0397 seconds for performing both pressure solving and velocity field projection. The same operations for the regular grid take 0.0374 seconds. The advection step takes slightly longer than for regular grids: 0.0065 seconds against 0.0042 seconds. While these numbers confirm that the performance of our technique is comparable to

the regular-grid one, our simulations are more accurate and our results are clearly superior (Figure 12).

Figure 10: Side view of the 3-D wind-tunnel simulation shown in Figure 1. Note the vortices behind the car.

5.1 Divergence-Free Velocity Fields and Lifting Forces

To verify that our projection technique for curvilinear grids produces divergence-free velocity fields, we computed the divergence for each cell of all curvilinear grids during various simulations after the projection step. In these experiments, the average cell divergence for the 2-D grids was 3.88×10^{-7}, while for the 3-D grid of the car in Figure 1 it was -5.13×10^{-9}. These numbers confirm the effectiveness of our mass-conservation technique.

Figure 13 shows the lift coefficients computed for an airfoil (NACA 6412) at maximum lift force for various angles of attack (ranging from -6 to 10 degrees). It compares the coefficients obtained using our technique, the variational method [BBB07a], and the traditional regular-grid approach against predictions for NACA 6412 at a Reynolds number of 50,000 [Air13]. The grid resolutions are the same used for the examples of Figure 12. The graph shows that the coefficients obtained with our technique are good approximations for the predicted values in the entire range of angles of attack. While the regular grid was expected to poorly approximate the predictions, it is somehow surprising to see that the variational approach largely diverges from the predictions for angles of attack in the range from -6 to 10 degrees. Their results also oscillate along the entire range.

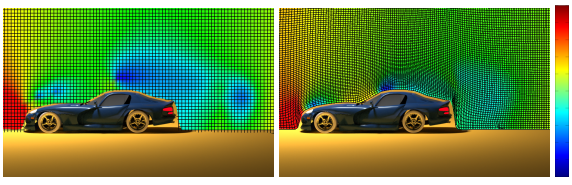

Figure 11: Cross sections of the pressure fields for one time step of a 3-D simulation using a regular grid (left) and our approach (right). Red means high pressure.

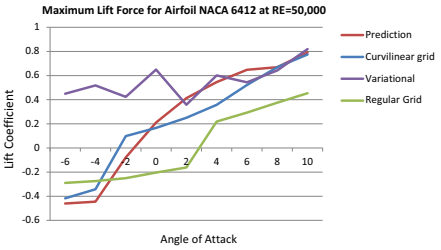

Figure 13: Comparison of lift coefficients at maximum lift force computed with various methods against the predictions for airfoil NACA 6412 at a Reynolds number of 50,000.

5.2 Discussion and Limitations

Our robust mass-conservation method (Section 3.3) produces oscillation-free velocity fields and supports a staggered-grid arrangement of variables. The cost of our method is $O(N)$: advection and velocity-projection steps touch each cell face once, and pressure solving is performed using a Multigrid method, which is $O(N)$.

We adopt a multi-step adaptive Runge-Kutta method to enforce that the semi-Langrangian path integrator will not skip cells during the advection phase. The adaptive-path integrator time-step δt is defined as $\delta t = min(\kappa, \Delta t)$, where Δt is the global time-step, $\kappa = min(\frac{h_1}{|\vec{u}_x|}, \frac{h_2}{|\vec{u}_y|}, \frac{h_3}{|\vec{u}_z|})$, and h_1, h_2 and h_3 are the local cell scale factors defined in Eq. (9). We integrate the particle's trajectory using δt until it reaches Δt, using the Runge-Kutta half-step method. Since curvilinear grids have cells with different sizes, the number of steps required for different cells may vary along the grid.

With this approach, we are able to relax the CFL condition on our curvilinear-grid method, by the same amount as standard regular grids. For overlapping grids, the CFL can be relaxed within certain limit: in our tests, we could reach a maximum CFL of 2.5. For higher CFL values, the information would travel faster than the algorithm updates the transfers between grids.

Our technique requires generating curvilinear-grid belts around the objects. These can be conveniently generated using a variety of available tools and techniques. For instance, Figure 7 illustrates a simple, fast, and effective method that consists of extruding a mesh along its normal vectors. All shown 2-D grids, including the illustration in Figure 7, were produced using *Gridgen* [Poi12], which uses this technique. Gridgen creates a mesh and optimizes its grid lines for orthogonality and to avoid foldovers, instantaneously.

Generating of 3-D grids for complex geometries is not as simple. Complex geometries must be decomposed in several parts, each part having its own extruded curvilinear grid. This kind of decomposition approach has been used in CFD for a long time, and software, documentation, and examples are available on-line [Nas13]. We implemented a simple 3-

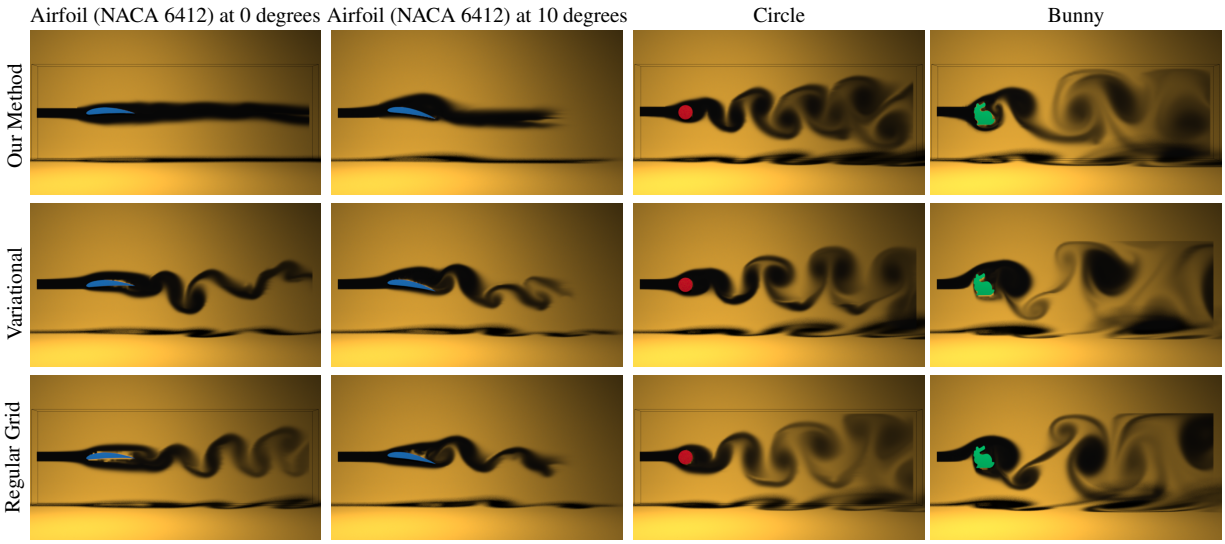

Figure 12: Comparison of our technique to the variational approach and to the regular grid method for a variety of 2-D examples: airfoil (NACA 6412), a circle, and a cross section of the Stanford bunny. The images show all methods at the same instant of the simulation.

Example	Advection time (s)		Press. + Proj. time (s)		Total time (s)		Number of cells	
	Regular	Our	Regular	Our	Regular	Our	Regular	Our
Airfoil 2-D	0.0041	0.0075	0.0387	0.0441	0.0497	0.0580	20,667	23,463
Circle 2-D	0.0042	0.0065	0.0374	0.0397	0.0481	0.0491	20,667	19,206
Bunny 2-D	0.0039	0.0044	0.0391	0.0368	0.0488	0.0451	20,667	17,061
Bunny & Circle 2-D	0.0042	0.0064	0.0387	0.0421	0.0465	0.0594	20,667	23,463
Bunny & Circle 2-D Dyn.		0.0081		0.0661		0.0771		27,201
Wind tunnel 3-D	0.9922	1.1720	1.327	1.4738	3.242	3.654	1,365,525	1,365,525

Table 1: Performance comparison between our approach and a regular-grid one using a CPU multigrid implementation.

D grid generation method that consists of NURBS fitting and extrusion to generate a single-blocked curvilinear grid. While this approach simplifies grid generation, it tends to smooth out some geometry details. It was used merely as a trade-off between accuracy and complexity of grid generation. Our approach is not yet suitable for dynamically deforming meshes, but it could be extended as in [BHS12].

6 Conclusions and Future Work

We presented an efficient approach for creating high-quality flow simulations based on curvilinear grids. Our method adapts and integrates in a single solution the best features scattered across several CFD methods: simple formulation, efficient evaluation, unconditional stability, accurate representation of boundary conditions, and support to a staggered-grid variable arrangement and overlapping grids. It can also be used with acceleration techniques for flow simulation, such as *coarse-grid projection* [LZF10].

Our approach is based on a fast unconditionally-stable semi-Lagrangian method that uses a domain transformation, and on a simple and robust mass-conservation method that produces oscillation-free velocity fields. It is easy to implement and its cost is nearly identical to the one of a regular-grid flow solver.

We have demonstrated the effectiveness of our approach through a series of 2-D and 3-D simulations and animations. Compared to approaches that try to improve the representation of boundary cells on regular grids [LGF04, RZF05, BBB07a], our method produces more accurate simulations and could potentially be used for interactive computation of drag and lifting forces on complex dynamic objects.

Our method can generate substantially more accurate results in several cases, although some trade-offs must be considered in order to treat complex 3-D geometries. We note that overlapping grids are an active research area in CFD. For instance, they are key for simulating aerodynamical properties of spacecrafts under extreme conditions. We expect that automatic grid-generation techniques be further de-

veloped in the upcoming years, facilitating the use of curvilinear grid in computer graphics.

References

[Air13] AIRFOILTOOLS: NACA 6412 (naca6412-il) xfoil prediction polar at re=50,000, 2013. http://airfoiltools.com/polar/details?polar=xf-naca6412-il-50000. Last access, May 2013. 8

[Bar84] BARR A. H.: Global and local deformations of solid primitives. *SIGGRAPH Comput. Graph. 18* (Jan 1984), 21–30. 4

[BBB07a] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3 (2007), 100. 1, 2, 7, 8, 9

[BBB07b] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling - project website, 2007. http://www.cs.ubc.ca/nest/imager/tr/2007/Batty_VariationalFluids/. Last access, May 2013. 7

[BG12] BELL N., GARLAND M.: Cusp - generic parallel algorithms for sparse matrix and graph computations., May 2012. http://code.google.com/p/cusp-library/. Last access, May 2013. 7

[BHS12] BANKS J. W., HENSHAW W. D., SCHWENDEMAN D. W.: Deforming composite grids for solving fluid structure problems. *J. Computational Physics 231*, 9 (2012), 3518 – 3547. 9

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. A K Peters, 2008. 2

[BXH10] BATTY C., XENOS S., HOUSTON B.: Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum 29*, 2 (2010), 695–704. 3

[CFL*07] CHENTANEZ N., FELDMAN B. E., LABELLE F., O'BRIEN J. F., SHEWCHUK J. R.: Liquid simulation on lattice-based tetrahedral meshes. In *Proc. 2007 ACM SIGGRAPH/Eurographics SCA* (2007), pp. 219–228. 3

[CG72] CONCUS P., GOLUB G. H.: *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations.* Tech. rep., Stanford, CA, USA, 1972. 2, 5

[Cho68] CHORIN A. J.: Numerical solutions of the Navier-Stokes equations. *Mathematics of computation 22*, 104 (1968), 745–762. 3

[DMYN08] DOBASHI Y., MATSUDA Y., YAMAMOTO T., NISHITA T.: A fast simulation method using overlapping grids for interactions between smoke and rigid objects. *Computer Graphics Forum 27*, 2 (2008), 477–486. 5, 6, 7

[EQYF13] ENGLISH R. E., QIU L., YU Y., FEDKIW R.: Chimera grids for water simulation. *SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2013). 3, 5, 6

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. SIGGRAPH '01* (2001), pp. 23–30. 2

[FOK05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M.: Animating gases with hybrid meshes. *ACM Trans. Graph. 24*, 3 (2005), 904–909. 2, 3

[FOKG05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M., GOKTEKIN T. G.: Fluids in deforming meshes. In *Proc. 2005 ACM SIGGRAPH/Eurographics SCA* (2005), pp. 255–259. 3

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proc. SIGGRAPH'01* (2001), pp. 15–22. 1

[HBW03] HOUSTON B., BOND C., WIEBE M.: A unified approach for modeling complex occlusions in fluid simulations. In *ACM SIGGRAPH 2003 Sketches & Appl.* (2003), p. 1. 2

[Hen05] HENSHAW W. D.: On multigrid for overlapping grids. *SIAM J. Sci. Comput. 26*, 5 (May 2005), 1547–1572. 6

[HS06] HENSHAW W. D., SCHWENDEMAN D. W.: Moving overlapping grids with adaptive mesh refinement for high-speed reactive and non-reactive flow. *J. Comput. Phys. 216*, 2 (Aug. 2006), 744–779. 2

[KC97] KARPIK S. R., CROCKETT S. R.: Semi-Lagrangian algorithm for two-dimensional advection-diffusion equation on curvilinear coordinate meshes. *Journal of Hydraulic Engineering 123* (May 1997), 389–401. 4

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3 (2006), 820–825. 3

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 457–462. 1, 2, 9

[LZF10] LENTINE M., ZHENG W., FEDKIW R.: A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Trans. Graph. 29* (July 2010), 114:1–114:9. 2, 9

[Nas13] NASA: Chimera grid tools user's manual., 2013. http://people.nas.nasa.gov/ rogers/cgt/doc/man.html. Last access, May. 2013. 8

[Poi12] POINTWISE: Pointwise gridgen - reliable cfd meshing software., 2012. http://www.pointwise.com/gridgen/. Last access, May 2013. 7, 8

[REN*04] RASMUSSEN N., ENRIGHT D., NGUYEN D., MARINO S., SUMNER N., GEIGER W., HOON S., FEDKIW R.: Directable photorealistic liquids. In *Proc. 2004 ACM SIGGRAPH/Eurographics SCA* (2004), SCA '04, pp. 193–202. 2

[Rob81] ROBERT A.: A stable numerical integration scheme for the primitive meteorological equations. *Atmosphere-Ocean 19*, 1 (1981), 35–46. 3

[RZF05] ROBLE D., ZAFAR N. B., FALT H.: Cartesian grid fluid simulation with irregular boundary voxels. In *ACM SIGGRAPH 2005 Sketches* (2005), SIGGRAPH '05, ACM. 1, 2, 9

[SDB83] STEGER J. L., DOUGHERTY F. C., BENEK J. A.: A chimera grid scheme. *Advances in Grid Generation 5* (1983). 2, 5

[SFK*08] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An unconditionally stable maccormack method. *J. Sci. Comput. 35* (June 2008), 350–371. 4, 7

[Sor02] SORLI K.: *A Review of Computational Strategies for Complex Geometry and Physics*. Tech. rep., NTNU, 2002. 1

[Sta99] STAM J.: Stable fluids. In *Proc. SIGGRAPH'99* (1999), pp. 121–128. 1, 3, 7

[Sta03] STAM J.: Flows on surfaces of arbitrary topology. In *Proc. SIGGRAPH 2003* (2003), pp. 724–731. 3

[SV91] SHYY W., VU T. C.: On the adoption of velocity variable and grid system for fluid flow computation in curvilinear coordinates. *J. Computational Physics 92*, 1 (1991), 82 – 105. 5

[WP00] WANG Z. J., PARTHASARATHY V.: A fully automated chimera methodology for multiple moving body problems. *International Journal for Numerical Methods in Fluids 33*, 7 (2000), 919–938. 5

[Wym01] WYMAN N.: State of the art in grid generation. CFD review, May 2001. http://www.cfdreview.com/article.pl?sid=01/04/28/2131215. Last access, May 2013. 3