Eurographics / ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

# Visual Simulation of Shockwaves

Jason Sewall    Nico Galoppo    Georgi Tsankov    Ming Lin[†]

University of North Carolina at Chapel Hill[‡]

**Abstract**

*We present an efficient method for visual simulations of shock phenomena in compressible, inviscid fluids. Our algorithm is derived from one class of the finite volume method especially designed for capturing shock propagation, but offers improved efficiency through physically-based simplification and adaptation for graphical rendering. Our technique is well suited for parallel implementation on multicore architectures and is also capable of handling complex, bidirectional object-shock interactions stably and robustly. We describe its applications to various visual effects, including explosion, sonic booms and turbulent flows.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism G.1.8 [Mathematics of Computing]: Partial Differential Equations

## 1. Introduction

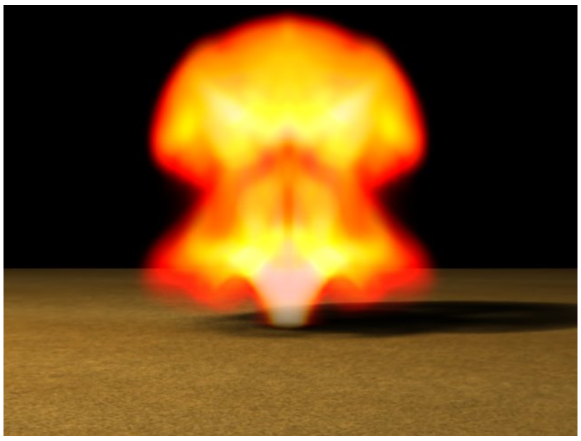
Recent developments in simulating natural phenomena have made it possible to incorporate stunning, realistic animations of complex natural scenes filled with flowing, bubbling, and burning fluids. Computer-animated and live-action films alike have made great use of these advances in modeling to recreate familiar and interesting effects. Notably, little investigation has been made on how to properly capture shocks and propagate discontinuities in visual simulation. These remarkable phenomena give rise to dramatic events such as explosions, turbulent flows, and sonic booms. Such effects are common in films and are notoriously difficult to handle with numerical methods. Additionally, many state-of-the-art simulation techniques do not fully take advantage of the kind of new, powerful hardware that is emerging; these algorithms are often not designed to handle large domains efficiently and many that are based on specially simplified formulations often are not applicable to phenomena occurring at large spatial scales.

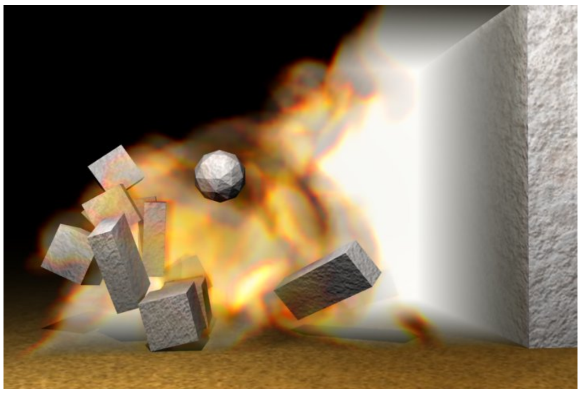
This paper presents a method for efficient simulations of nonlinear, compressible gas dynamics and how it may be best utilized to generate visually interesting, plausible animations. Many natural phenomena are nonlinear but can often be reasonably approximated through linearization; one example is elasticity. The equations of fluid motion are not generally suitable for linearization — waves crashing on the beach, curling smoke, and surging shockwaves all arise from the nonlinear characteristics of the system. To solve these highly nonlinear equations in a reasonable amount of time, numerical methods typically discretize simplifications of the true equations that still capture the nonlinearity of the system.

Furthermore, shocks that arise in problems of gas dynamics themselves present a numerical challenge; a shock is a region of rapid spatial variation in a small interval that propagates with tremendous speed — the blast wave that emanates from an explosion or the bow shock that forms around a supersonic projectile are some examples of these phenomena. These have a striking effect on the fluid motion but are very difficult to simulate properly with traditional numerical methods; the scale of motion we desire to capture (namely the space the shock traverses) is at odds with the need to represent the shock itself. Many numerical techniques behave poorly or fail completely in the presence of discontinuous solutions — to simulate shocks with such methods, the resolution of the discretization must be high enough for the shock to appear as a smooth transition, and thus can be prohibitively expensive to compute.

Physically correct methods for shockwave modeling focus less on conventional metrics of accuracy (such as order of convergence) and emphasize the ability to propagate discontinuities stably and with minimal diffusion. Specifically, techniques based on the finite volume method (FVM) have been developed that handle discontinuities well and al-

[†] e-mail: {sewall, nico, gtsankov, lin}@cs.unc.edu
[‡] Chapel Hill, NC, 27599-3175 USA


(a) A mushroom cloud generated by our method


(b) A stack of rigid bodies knocked over by a shock

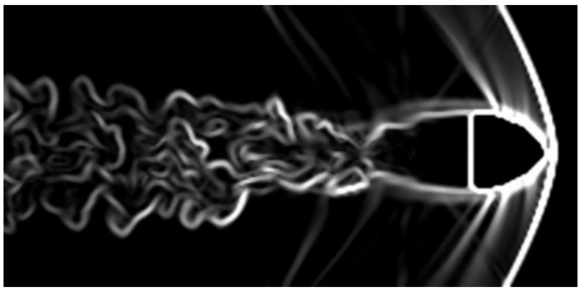

(c) A bow shock and turbulence formed by the passage of a supersonic bullet

**Figure 1:** *Dramatic scenes generated by our method*

low for relatively coarse grids to capture shock behavior. Our method, through judicious simplification and application, adapts and improves the efficiency of a class of FVM techniques designed to capture shocks on coarse grids efficiently.

We have demonstrated our method on generating animations of complex fluid motion, including chambered explosions, nuclear detonations, and the turbulence and bow shock around a supersonic projectile (see Fig. 1). Our method is also able to describe the interaction of coupled fluids and objects; we demonstrate shockwaves knocking over stacked objects and blowing a brick house to pieces, as well as the effects of an explosion within a tower of heavy blocks. Our method is able to considerably reduce the computational complexity of these highly complex effects to the level comparable to existing fluid animation techniques in graphical simulation. Furthermore, our method is naturally amenable to parallelization on multicore architectures.

## 2. Previous Work

In addition to many decades of research in computational fluid dynamics, there is a considerable amount of literature on the modeling of fluid phenomena in computer graphics [Sta99, LGF04, APKG07, CFL*07, BBB07]. The seminal works of Foster and Metaxas [FM96], Stam [Sta99], and Foster and Fedkiw [FF01] on incompressible fluid simulation were among the first to examine this topic for visual simulation. For a detailed explanation of the numerical methods and the mathematics behind them, we refer the readers to a recent set of comprehensive course notes on fluid simulation [BFMF06].

Recent work on fluid simulation based on the finite volume methods has been discretized on irregular grids [FOK05, ETK*07, WBOL07]. Subsequent improvements [KFCO06, CGFO06, CFL*07] to these methods have combined the best features of the initial publications to achieve impressive results.

The finite volume method has received much attention from the aeronautics community; our technique uses numerical Riemann solvers based on the work done by Roe [Roe81], van Leer [vL77], and others. For a superb introduction to the topic of the finite volume method and Riemann solvers, see [Lev02].

The problem of describing the evolution of shocks — known as “shock capturing” — has been addressed from a variety of directions. Our work follows the vein of *Riemann solver-based* approaches that strive to treat areas with and without shocks with the same numerical technique. Another family of approaches, generally known as *front-tracking* methods, uses standard solvers in areas away from shocks and explicitly models shocks as evolving surfaces in the domain. Front-tracking approaches have been successful, but are extremely complicated for two- and three-dimensional simulations and have difficulty handling situations where multiple shocks interact. The survey of Fedkiw et al. [FSS03] gives a good overview of the topic.

Relatively little work in computer graphics has utilized the Euler equations — that is, the compressible, inviscid simplification of the Navier-Stokes system of equations — all of the aforementioned methods from the graphics community are based on an incompressible simplification of the equations. Yngve et al. [YOH00] present a method for high-energy, compressible fluid simulation based on finite differences, which they use to simulate explosions and their secondary effects. Sewall et al. [SMML07] use a method similar to finite volume on irregular grids to simulate compressible flow.

Feldman et al. [FOA03] simulate combustive phenomena based on an incompressible model of flow with additional density tracers, and Selle et al. [SRF05] present an approach that generates what they describe as "rolling explosions". Like Feldman et al., they use an incompressible model of fluid, which precludes the presence of shocks. Our approach aims to model phenomena similar to those addressed by Yngve et al [YOH00]. The greater fidelity and higher efficiency afforded by our method opens up a wide range of new applications of these phenomena to visual effects.

Work done by Müller et al. [MCG03] and Adams et al. [APKG07] on particle-based fluid with the Smooth Particle Hydrodynamics (SPH) method strives to represent incompressible flow. The natural tendency of the space between particles to expand and collapse suggests potential future application to compressible phenomena.

Several methods have addressed the considerable challenge of coupling fluids to objects. Geneveaux et al. [GHD03] suggest an explicit method for the bidirectional coupling of grid-based fluids to solid bodies using a particle representation of the surface. Carlson et al. [CMT04] use distributed Lagrange multipliers to achieve stable fluid-object coupling and Guendelman et al. [GSLF05] describe how to handle the interaction of infinitely thin shells with fluids. Chentanez et al. [CGFO06] use an implicitly-coupled model of fluid and elastic bodies to obtain stable interactions. Batty et al. [BBB07] use variational principles to develop a simple extension to the pressure projection step to achieve stable two-way coupling in incompressible fluids. We achieve bidirectional coupling through voxelization (as with [CMT04] and [BBB07]). We use simple modifications to the Riemann solvers on boundary interfaces to affect the interaction.

There has been some work on simulating the effects of blast waves through analytical models of blast propagation. Mazarak et al. [MMA99] used an expanding ball to determine forces on bodies to fracture or propel them. Neff and Fiume [NF99] use similar analytic models of blast waves to fracture objects and are also unable to generate the aforementioned effects of shock dynamics. These approaches are typically quite fast, but their extremely simple model of blast dynamics does not allow for the effects of shock-object interaction — notably reflection and vortex shedding — nor do they have the ability to visualize the blast itself. Neff and Fiume [NF99] use similar analytic models of blast waves to fracture objects and are also unable to generate the aforementioned effects of shock dynamics.

## 3. Method

The key challenge is to simulate shockwave and compressible-gas dynamics by designing a practical numerical method that can stably handle moving boundary conditions in three-dimensional space and is efficient enough to be used in a visual simulation production pipeline. We present a basic introduction to the finite volume method and refer the readers to [Lev02] for more detail.

### 3.1. Conservation laws

We seek solutions to the Euler equations of gas dynamics. These equations form a *hyperbolic conservation law*, the general, three dimensional form of which is:

$$\mathbf{q}_t + \mathbf{F}(\mathbf{q})_x + \mathbf{G}(\mathbf{q})_y + \mathbf{H}(\mathbf{q})_z = 0 \tag{1}$$

where subscripts indicate partial differentiation.

Here $\mathbf{q}$ is the vector of unknowns and $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$ are vector-valued *flux functions* specific to each conservation law. A conservation law states that a quantity of unknowns $\mathbf{q}$ over an arbitrary domain $S$ changes in time due only to the flux across the boundary $\partial S$ of the domain.

#### 3.1.1. Integral form

The derivatives found in partial differential equations such as Eq. (1) are not defined around discontinuities; to capture them properly we use an integral form of the equations.

In one dimension, consider an interval $[a,b]$; the change in $\mathbf{q}$ over that interval is due to the flux at $a$ and the flux at $b$. More formally,

$$\frac{d}{dt}\int_a^b \mathbf{q}(x,t)\,dx = \mathbf{F}(\mathbf{q}(a,t)) - \mathbf{F}(\mathbf{q}(b,t)) \tag{2}$$

where $\mathbf{F}$ is a flux function such as $\mathbf{F}$, $\mathbf{G}$, or $\mathbf{H}$ from Eq. (1) and we follow the convention that 'positive flux' is left-going and 'negative flux' right-going.

### 3.2. The finite volume method

The finite volume method (FVM) on regular grids follows directly from Eq. (2); the presentation here is for scalar equations in one dimension with scalar unknowns $q$ and scalar fluxes $f$, but the formulae for systems of equations in multiple dimensions are straightforward extensions of these.

We discretize the spatial interval $[a,b]$ into $m$ intervals ("cells") of equal size $\Delta x = \frac{b-a}{m}$. For each time $t_n$, we have $m$ quantities $Q_i^n$ defined as the average value of $q$ over the cell:

$$Q_i^n = \frac{1}{\Delta x}\int_{\chi_i^l}^{\chi_i^r} q\,(x,t_n)\,dx \tag{3}$$

where we have $\chi_i^l = a + i\Delta x$, and $\chi_i^r = \chi_i^l + \Delta x$ as the positions of the cell boundaries. Observe that $\chi_{i-1}^r = \chi_i^l$.

We apply Eq. (2) to each of the intervals $i$

$$\frac{d}{dt}\int_{\chi_i^l}^{\chi_i^r} q(x,t)\,dx = f\Big(q(\chi_i^l,t)\Big) - f\Big(q(\chi_i^r,t)\Big) \tag{4}$$

and integrate Eq. (4) from $t_n$ to $t_{n+1}$

$$\int_{\chi_i^l}^{\chi_i^r} q\,(x,t_{n+1})\,dx - \int_{\chi_i^l}^{\chi_i^r} q\,(x,t_n)\,dx = \int_{t_n}^{t_{n+1}} f\Big(q(\chi_i^l,t)\Big) - f\Big(q(\chi_i^r,t)\Big)\,dt \tag{5}$$

Observe that we can substitute Eq. (3) if we divide Eq. (5) by $\Delta x$.

$$Q_i^{n+1} - Q_i^n = \frac{1}{\Delta x}\int_{t_n}^{t_{n+1}} f\Big(q(\chi_i^l,t)\Big) - f\Big(q(\chi_i^r,t)\Big)\,dt \tag{6}$$

The right-hand side of this equation is a *flux difference* that cannot generally be evaluated exactly; we approximate the integrals with averages over the cell interfaces from $[t_n, t_{n+1}]$:

$$F^n_{\chi^l_i} \approx \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} f\left(q(\chi^l_i, t)\right) dt \qquad (7)$$

Substituting Eq. (7) into Eq. (6), we obtain the most basic FVM update scheme:

$$Q^{n+1}_i = Q^n_i - \frac{\Delta t}{\Delta x}\left(F^n_{\chi^l_i} - F^n_{\chi^r_i}\right) \qquad (8)$$

This scheme is first-order and is subject to the numerical viscosity typical of first-order methods. Second-order schemes such as the Law-Wendroff scheme [LW60] can be employed with comparable computation effort; we complement this with a *flux limiter*, which minimizes diffusive and dispersive artifacts. We have used the MC limiter of van Leer [vL77] in our method.

## 3.3. The Riemann problem

According to Eq. (7), the flux at $\mathbf{F}_{i-\frac{1}{2}}$ is dependent on the state $\mathbf{Q}_{i-\frac{1}{2}}$ at the interface between $\mathbf{Q}_{i-1}$ and $\mathbf{Q}_i$ over the interval $(t_n, t_{n+1})$; thus we must determine the value at this interface as it evolves in time. $\mathbf{Q}_i$ here is the vector version of the discrete unknowns first introduced in 3.2.

Given the initial data:

$$\mathbf{Q}(x,0) = \begin{cases} \mathbf{Q}_l, & x < 0 \\ \mathbf{Q}_r, & x \geq 0 \end{cases} \qquad (9)$$

we wish to solve for $\mathbf{Q}(x,t)$ for $t > 0$ subject to the governing equations. This formulation is known as the *Riemann problem* for the governing equations; the resulting $\mathbf{Q}(0,t)$ obtained can then be used to compute the flux at the cell interface.

### 3.3.1. Riemann problem for linear systems

Let us consider linear, constant-coefficient (but not necessary scalar) hyperbolic conservation laws, i.e. Eq. (1) where the flux function $\mathbf{F}$ takes the form $\mathbf{F}(\mathbf{q}) = \mathbf{A}\mathbf{q}$, where $\mathbf{A}$ is a *flux matrix*. (Assume that the other flux functions $\mathbf{G}$, and $\mathbf{H}$ are of the same form).

Such a system of order $n$ can be diagonalized into $n$ decoupled equations $\mathbf{Q}^+_t + \Lambda \mathbf{Q}^+_x = 0$, where $\mathbf{Q}^+ = \mathbf{R}^{-1}\mathbf{Q}$. Here $\mathbf{R}$ is the matrix of right eigenvectors of $\mathbf{A}$, and $\Lambda$ is a diagonal matrix of the eigenvalues of $\mathbf{A}$ satisfying $\mathbf{A} = \mathbf{R}\Lambda\mathbf{R}^{-1}$.

The solution to the Riemann problem for these equations is given by $n$ weighted eigenvectors $W_i = \alpha_i \mathbf{r}_i$ (also known as *waves*) propagating with speeds $\lambda_i$, the corresponding eigenvalues.

The waves $W_i$ are determined by projecting the jump in the initial states $\Delta\mathbf{Q} = \mathbf{Q_r} - \mathbf{Q_l}$ onto the space formed by the eigenvectors of the system:

$$\sum_i W_i = \sum_i \alpha_i \mathbf{r_i} = \Delta\mathbf{Q} \qquad (10)$$

$$\mathbf{R}\mathbf{a} = \Delta\mathbf{Q} \qquad (11)$$

$$\mathbf{a} = \mathbf{R}^{-1}\Delta\mathbf{Q} \qquad (12)$$

where $\mathbf{a} = [\alpha_0, \alpha_1, \ldots, \alpha_{n-1}]^{\mathrm{T}}$

The waves define $k$ intermediate states $\mathbf{Q}^{*_i} = \mathbf{Q}_l + \sum_{j=0}^{i} W_j$, and the solution to the Riemann problem is therefore the piecewise-constant function

$$\mathbf{Q}(x,t) = \begin{cases} \mathbf{Q}_l, & x < \lambda_1 t \\ \vdots & \vdots \\ \mathbf{Q}^{*_i}, & \lambda_i t < x \leq \lambda_{i+1} t \\ \vdots & \vdots \\ \mathbf{Q}_r, & x \geq \lambda_n t \end{cases} \qquad (13)$$

### 3.3.2. The Riemann problem for nonlinear systems

For nonlinear systems such as the Euler equations, the wave structure of the solution is much more complicated and costly to compute — typically, iterative root-finding methods must be employed at each cell interface to determine the intermediate states $\mathbf{Q}^{*_i}$.

However, it is often possible to obtain good results by approximately solving the Riemann problem; through linearizations of the flux evaluated at carefully chosen states, we can obtain solutions that fit Eq. (13). Such approximate Riemann solvers must be used with care, as they can often produce non-physical solutions. We discuss the applicability of these solvers and how these undesirable conditions can be addressed in Sec. 3.4.1.

### 3.3.3. Upwinding flux splitting

The basic FVM Eq. (8) update scheme developed in Sec. 3.2 is not able to stably handle hyperbolic systems; we need to modify it to obey the principle of *upwinding*. We must take care to ensure that waves traveling in the positive direction use information from the *negative* direction.

Rather than use Eq. (8) to compute cell updates, we employ a scheme

$$Q^{n+1}_i = Q^n_i - \frac{\Delta t}{\Delta x}\left(F^{n+}_{\chi^r_i} + F^{n-}_{\chi^l_i}\right) \qquad (14)$$

where $F^{n-}_{\chi^r_i}$ is the part of the flux $F^n_{\chi^r_i}$ traveling in the negative direction and $F^{n+}_{\chi^r_i}$ the part traveling in the positive direction.

The waves $W_i$ and speeds $\lambda_i$ from the solution to a Riemann problem at $\chi^r_i$ is then

$$F^{n-}_{\chi^r_i} = \sum_{c=0}^{j} \lambda_c W_c \qquad F^{n+}_{\chi^r_i} = \sum_{c=j}^{k} \lambda_c W_c \qquad (15)$$

Where $\ldots < \lambda_j < 0 < \lambda_{j+1} < \ldots$; waves traveling with negative speeds are added to $F^{n-}$ while those traveling with positive speed are added to $F^{n+}$.

### 3.3.4. Solution procedure

Given cell values $\mathbf{Q}^n$ for time $t_n$, a timestep is performed as follows to compute $\mathbf{Q}^{n+1}$:

1. For each interface between cells, compute the wavespeeds $\lambda_i$ and fluxes $F_i^n$ by solving the Riemann problem at that interface (described in Sec. 3.4.2)
2. Find the wavespeed with largest magnitude from $|\lambda_i|$ to compute timestep length $\Delta t$ as described in Sec. 3.4.4.
3. For each cell $i$, advance to next time $\mathbf{Q}^{n+1}$ using the fluxes $F^n$ at its neighboring interfaces $\chi_i^l$, $\chi_i^r$ using Eq. (14).

For three-dimensional problems (see Sec. 3.4.4), we must compute three fluxes for each cell in the domain; solving the Riemann problems in step 1 becomes the computational bottleneck for non-trivial systems of equations. While expensive to obtain, carefully calculated fluxes are the key to handling discontinuous solutions on a coarse grid. Next, we describe what the Riemann problem is and how it can be used to compute flux between cells.

## 3.4. The Euler equations

We are interested in studying the motion of a compressible gas; the natural choice is the Euler system of equations. The simplification of Navier-Stokes that omits viscous terms results in this nonlinear hyperbolic system of conservation laws. The omission of viscosity is a reasonable one to make for many physical problems in gas dynamics, just as the incompressible simplification of Navier-Stokes frequently used in graphics is reasonable for liquid simulation.

The Euler equations in conservation form (see Eq. (1)) are

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, \quad \mathbf{F}(\mathbf{q}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (E+p)u \end{bmatrix}$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ (E+p)v \end{bmatrix}, \quad \mathbf{H}(\mathbf{q}) = \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ (E+p)w \end{bmatrix} \tag{16}$$

Here $\rho$ is density, $u$, $v$, and $w$ the components of velocity, $p$ the pressure, and $E$ the total energy. An additional *equation of state* completes the system

$$E = \frac{p}{\gamma - 1} + \frac{\rho}{2}\left(u^2 + v^2 + w^2\right) \tag{17}$$

where $\gamma$ is the adiabatic exponent of the fluid — typically 1.4 for air. It should be noted that for solutions to be physically valid, $\rho$, $p$, and $E$ must all be strictly greater than zero.

### 3.4.1. Approximate Riemann solutions

As discussed in Sec. 3.3.2, computing the exact solution to the Riemann problem for nonlinear systems such as the Euler equations is prohibitively expensive for practical problems. Suitably approximated solutions to the Riemann problem are often able to achieve acceptable results for a fraction of the cost of solving them exactly.

We would like to apply the method for solving Riemann problems for linear systems presented in Sec. 3.3.1 to nonlinear problems; to this end we desire a matrix $\mathbf{A}$ such that $\mathbf{A}$ approximates $\mathbf{F}'(\mathbf{Q})$; here $\mathbf{F}'(\mathbf{Q})$ is the Jacobian of $\mathbf{F}$ as seen in the *quasilinear* form of the conservation law. This is simply the chain rule applied to (1): $\mathbf{Q}_t - \mathbf{F}(\mathbf{Q})_x = \mathbf{Q}_t - \mathbf{F}'(\mathbf{Q})\mathbf{Q}_x = 0$.

In a seminal paper, Roe [Roe81] presented a simple method for approximating $\mathbf{F}'(\mathbf{Q})$ that preserves important conditions of the system, and it is this method that we have adapted for our solver. Roe's method uses a flux matrix $\mathbf{A}$ that is $\mathbf{F}'(\bar{\mathbf{Q}})$ evaluated at a specially chosen state $\bar{\mathbf{Q}}$ given $\mathbf{Q}_l$ and $\mathbf{Q}_r$ — this state has come to be known as the *Roe average state*.

**Eigenvectors and eigenvalues of the flux Jacobian** The eigenvectors of the Jacobian $\mathbf{F}'(\mathbf{Q})$ give the waves necessary to compute the intermediate states as in Sec. 3.3.1, and its eigenvalues give the characteristic speeds $\lambda_i$ with which these waves propagate. The eigenvalues of the flux Jacobian $\mathbf{F}'$ as computed from (16) are:

$$\lambda_{0\ldots4} = (u-c, u, u, u, u+c) \tag{18}$$

and the corresponding eigenvectors are:

$$\mathbf{r}_1 = \begin{bmatrix} 1 \\ u-c \\ v \\ w \\ H-uc \end{bmatrix} \quad \mathbf{r}_2 = \begin{bmatrix} 1 \\ u \\ v \\ w \\ \frac{1}{2}(u^2+v^2+w^2) \end{bmatrix}$$

$$\mathbf{r}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ v \end{bmatrix} \quad \mathbf{r}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ w \end{bmatrix} \quad \mathbf{r}_4 = \begin{bmatrix} 1 \\ u+c \\ v \\ w \\ H+uc \end{bmatrix} \tag{19}$$

Here $c = \sqrt{\frac{\gamma p}{\rho}}$ is the speed of sound and $H = \frac{E+p}{\rho}$ the total specific enthalpy. We have given only the eigenvalues and eigenvectors for $\mathbf{F}'$, but those for the Jacobians of the other flux functions $\mathbf{G}'$ and $\mathbf{H}'$ have similar structure.

**Roe average state** Given two states $\mathbf{Q}_l = [\rho_l, u_l, v_l, w_l, E_l]$ and $\mathbf{Q}_r = [\rho_r, u_r, v_r, w_r, E_r]$, the Roe average is

$$\bar{\mathbf{Q}} = [\bar{\rho}, \bar{u}, \bar{v}, \bar{w}, \bar{H}]^{\mathrm{T}} \qquad \bar{\rho} = \frac{\rho_l + \rho_r}{2} \tag{20}$$

$$\bar{u} = \frac{\sqrt{\rho_l}u_l + \sqrt{\rho_r}u_r}{\sqrt{\rho_l} + \sqrt{\rho_r}} \quad \bar{v} = \frac{\sqrt{\rho_l}v_l + \sqrt{\rho_r}v_r}{\sqrt{\rho_l} + \sqrt{\rho_r}} \tag{21}$$

$$\bar{w} = \frac{\sqrt{\rho_l}w_l + \sqrt{\rho_r}w_r}{\sqrt{\rho_l} + \sqrt{\rho_r}} \quad \bar{H} = \frac{\frac{E_l + p_l}{\sqrt{\rho_l}} + \frac{E_r + p_r}{\sqrt{\rho_r}}}{\sqrt{\rho_l} + \sqrt{\rho_r}} \tag{22}$$

The specific variables shown here (in contrast to the conservative variables given in (16)) appear because they are precisely what is necessary to evaluate the eigenvalues Eq. (18) and eigenvectors Eq. (19) and obtain the waves and speeds for a given Riemann problem.

This average state has attractive properties when considering the structure of the Riemann problem; were we to choose a simple arithmetic average of the quantities at $\mathbf{Q}_l$ and $\mathbf{Q}_r$, the resulting eigenvectors may fail to be distinct and the solution fail entirely. The criteria behind this particular choice of average are explained in detail in [Roe81] and [Lev02].

**Enforcing physicality** Using the Roe average state Eq. (20) to approximately solve the Riemann problem is significantly faster then computing the exact solution to the Riemann problem, but the solver is known to generate nonphysical states for certain inputs $\mathbf{Q}_l$ and $\mathbf{Q}_r$. While the exact solution to the Riemann problem could be computed to obtain the physically valid intermediate state, this is unnecessary and overly expensive for visual simulation. When the approximate Riemann solver produces invalid states, we apply slight corrections to enforce physicality. We clamp $\rho$ and $p$ to be no less than 0.05 — in the case of $p$, this entails adjusting $E$ according to Eq. (17).

For example, the fluid-rigid body simulations illustrated in Figs. 1(b), 5, 2, and 3 demonstrate plausible motion, and would not be possible using a simple approximate Riemann solver without these corrections.

### 3.4.2. Riemann solver for Euler equations

We have developed the theory of Riemann solvers for the Euler equations sufficiently to present the procedure for computing the Riemann solution at interface given left and right states $\mathbf{Q}_l$ and $\mathbf{Q}_r$.

1. Compute Roe average $\bar{\mathbf{Q}}$ using Eq. (20)
2. Make $\bar{\mathbf{Q}}$ physically valid if needed, as per Sec. 3.4.1
3. Compute wavespeeds $\lambda_i$ using to Eq. (18)
4. Compute eigenvectors $\mathbf{r}_i$ using Eq. (19)
5. Project $\Delta\mathbf{Q}$ onto eigenspace by computing the wave coefficients $\alpha_i$ and waves $W_i$ using Eq. (12)
6. Compute left and right fluctuations $F^{n\pm}$ using Eq. (15)

### 3.4.3. Boundary conditions

We apply boundary conditions where needed through modified Riemann solvers; these do not solve for the flux at an interface due to two adjacent cells; we compute a 'ghost' intermediate state at the interface to determine these fluxes. In practice, we have found three types of boundary conditions useful:

**Free-slip**: This common boundary condition simply states that the component of flow normal to the interface is zero. We obtain this by modifying the Roe average Eq. (20) used in the cell to have zero velocity in the component normal to the boundary; thus $\bar{u}$ on a no-slip boundary normal to the $x$-direction is set to zero. Other components of the intermediate state $\bar{\mathbf{Q}}$ are simply treated as though $\mathbf{Q_l}$ were equal to $\mathbf{Q_r}$.

**Velocity**: This is a generalization of free-slip boundary conditions; rather than enforce zero velocity along an interface, some user-specified velocity is imposed as the intermediate component of velocity in the appropriate direction. Other components are treated as though the two adjacent cells were identical except for the energy $E$; given an imposed velocity $\bar{u}$ and the same component of velocity in the adjacent cell $u_r$, the velocity in the ghost cell is $u_L = 2\bar{u}(\bar{u}-u_r)$. Due to this difference in velocity, the energy in the ghost cell is not equal to its neighbor and is adjusted with Eq. 17.

**Absorbing**: It is often desirable to perform simulations where outgoing waves are simply absorbed rather than reflected; the computational domain behaves as if it were suspended in an infinite passive medium. At these interfaces, the fluxes in the Riemann problem are simply set to zero.

### 3.4.4. Dimensional splitting

The discussion so far has been limited to one dimension — our equations Eq. (16) are three-dimensional, but the solution procedure in Sec. 3.3.4 performs updates in only a single dimension.

To solve three-dimensional problems, we perform *dimensional splitting*. To advance from time $t_n$ to $t_{n+1}$, we make sub-step "passes" of a one-dimensional solver in each direction — first using the flux function $\mathbf{F}$ along $x$ for all rows of constant $y$ and $z$, then using $\mathbf{G}$ along $y$ for all rows of constant $x$ and $z$, and finally using $\mathbf{H}$ along $z$ for all rows of constant $x$ and $y$.

This approach allows us to apply the one-dimensional techniques previously described here in a straightforward manner; however, we must address how best to choose the timestep to take over the three passes.

**Choosing a timestep** The timestep size $\Delta t$ that we are able to take while advancing the solution with Eq. (8) is limited by the maximum characteristic speed $\lambda_{\max}$ from Eq. (18) in the solution we are updating, as per the Courant-Friedrichs-Lewy (CFL) condition [CFL28]. For simulation in a single dimension, the procedure in Sec. 3.3.4 works perfectly — we compute the solution to all Riemann problems in the domain, which gives us the maximum characteristic speed, which we use to compute the timestep $\Delta t = \frac{\Delta x}{\lambda_{\max}}$. With dimensional splitting, we are not able to compute the maximum speed in dimension $y$ prior to advancing the solution in $x$ with some previously chosen $\Delta t$; the maximum speed in $y$ depends the results of the $x$-pass and is not generally equal to the $\lambda_{\max}$ from the $x$ pass.

There are several ways to address this problem — for example, we could adopt a guess-and-check approach of estimating a timestep, advancing the solution with it, checking to see if it satisfies the CFL condition based on the maximum speed of the next level, and rewinding the whole computation if not, but this would be prohibitively expensive.

We take the very simple approach of always advancing the solution in a dimension with the largest timestep that satisfies the CFL condition in that dimension. This method clearly has effects on the solution; effectively, the grid is 'warped' over a timestep based on the ratios of maximum speeds in each dimension. However, we have found these effects to be negligible in the simulations we have run, even in cases where the flow (and therefore $\lambda_{\max}$) is highly biased along a single dimension (see for example Figs. 1(b), 1(c), and 5).

Our approach has an advantage over other methods and is particularly desirable for visual simulation; each dimension is advanced according to the chosen CFL number of the simulation. No dimension is forced to take a timestep at a low CFL number because of other, higher speed dimensions. This technique helps reduce the numerical artifacts that frequently plague visual simulations of natural phenomena.

### 3.5. Fluid-object interaction

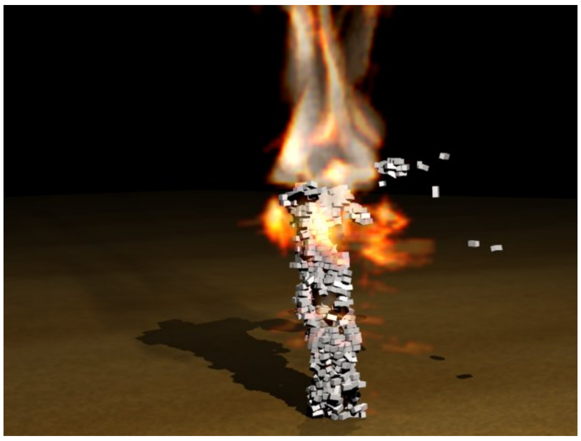


**Figure 2:** *Tower (without cap) blown apart by internal blast*

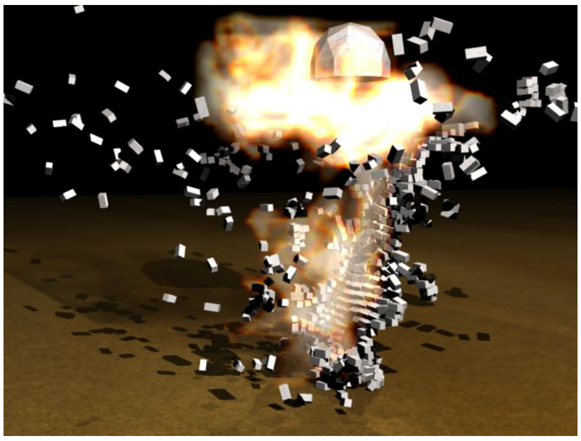


**Figure 3:** *Tower (with cap) blown apart by internal blast*

We employ a method for bi-directional fluid-object coupling that is simple, stable, and efficient. At each timestep, solid objects are voxelized onto the grid and cells occupied by solids marked as such.

To capture the objects' effect on the fluid, we use the aforementioned free-slip modification to the Riemann solver along the boundary (in Sec. 3.4.3). This solver ensures that incoming waves are reflected off of solid bodies and enables effects like those seen in Figs. 1(b), 5, 2, 3, and 4; these demonstrate the effects of the solids in the scene on the flow.

The force exerted by the fluid on the objects is obtained by multiplying the pressures at each incident cell by the interface's normal direction and applying the resulting force to the object. This simple technique is responsible for the forces buffeting the objects in Figs. 1(b), 5, 2, and 3.

Any rigid body simulator is suitable for use with our method; we have used the Bullet collision and dynamics engine [bul] because of its completeness and availability. Our voxelization is a simple custom tool based on triangle-grid intersections.

Considerable work [CMT04,CGFO06,BBB07] has been done to achieve stable fluid-solid interactions in the past, but these methods have focused on the interaction of rigid bodies with incompressible fluids. Stability problems frequently arise in such situations because of the differing needs of the rigid body dynamics and the fluid simulator; the implicit solver for incompressible fluid simulation generally takes large timesteps, which can result in a loosely-coupled, unstable simulation when rigid bodies are handled naïvely. Our method naturally takes many small, inexpensive timesteps to advance the solution; this allows tighter communication between the rigid body and fluid simulators and results in a more stable interaction.

## 4. Results

We have implemented and tested our algorithm on several challenging scenarios. In this section, we first show some demonstrations of our algorithm, then describe our rendering methods, and finally discuss timing and parallelization results.

### 4.1. Applications

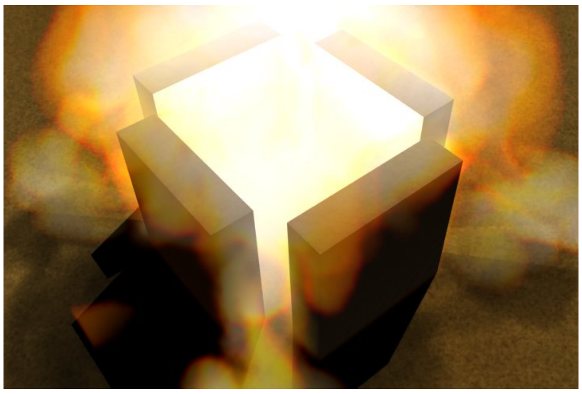


**Figure 4:** *An explosion in a confined space*

We have constructed a number of scenarios that demonstrate the ability of our method to simulate visually interesting phenomena. The first segment of supplementary video is a two-dimensional simulation demonstrating vortex shedding — a traveling shock crosses a sharp obstacle and a powerful vortex forms in its wake. Further reflections of the shocks create new vortices which combine and travel around the domain.

Fig. 1(a) shows a mushroom cloud formed in the aftermath of a nuclear explosion; a low-density, high-temperature region left by the expanding shock is forced upwards by the pressure gradient caused by gravity; as it rises, the region expands and curls downward, forming a distinctive mushroom shape.

Fig. 1(b) demonstrates our method's ability to interact

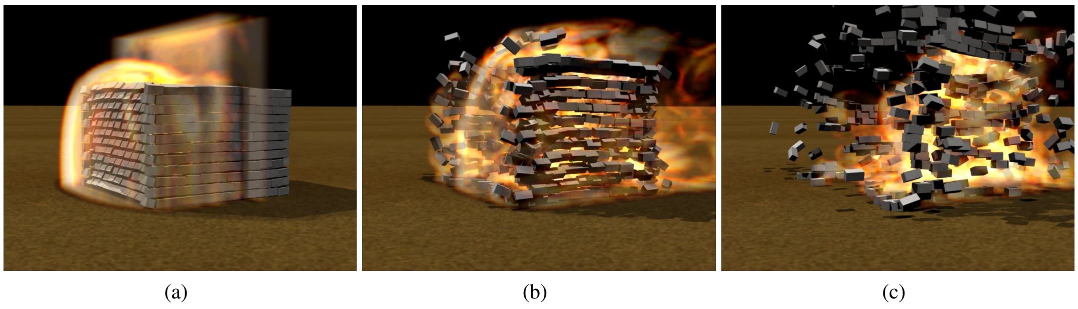


**Figure 5:** *Rigid body-fluid interaction*

with moving boundary conditions; the stack of rigid bodies in this scene are bidirectionally coupled to the fluid. A traveling shock topples them, reflects off a nearby wall, and rebounds on the objects, throwing them away. The bodies' force upon the fluid creates vorticial patterns in the gas.

Fig. 1(c) shows a two-dimensional slice of a three-dimensional simulation of a projectile traveling faster than the speed of sound. The bow shock ahead of the body is typical of this type of rounded object and the rarefaction region behind the projectile creates a twisting trail of turbulence.

Fig. 2 and fig. 3 are similar; in each, a cylindrical tower of 600 bricks is toppled by an explosion from within. Fig. 2 has no cap; the explosion forces nearly all of the air out of the cylinder as it bursts out of the top. The low-pressure area formed inside the cylinder causes it to collapse in upon itself while the force of the explosion venting from the top of the structure send bricks flying. Fig. 3 has a very heavy cap atop it; the explosive force cannot escape so easily and is partially reflected back into the structure, forcing a hole in the base and blowing out bricks near the top.

Fig. 4 shows an explosion occurring in an enclosed area; the force of the explosion forces air through the small openings in the chamber and creates high-density, turbulent tendrils.

Fig. 5 shows a series of frames from a simulation where a "house" made of 480 concrete bricks is struck by a powerful shock, causing the bricks to fly in all directions. The bricks shape and reflect the shock as it propagates through the scene.

Fig. 6 is a visual recreation of the first moments of the detonation of the first nuclear bomb 'Trinity'. The glossy "bubble" around the explosion is the expanding shockfront; the heat at the interface is such that light traveling through the region is dramatically refracted. Inside the shock, dust and flame are rising with a bright glow.

### 4.2. Rendering

Our three-dimensional demonstrations were modeled in Blender [ble] and rendered with the V-Ray raytracer [vra]; the visualization of fluid effects in 3D were handled by our Monte Carlo volume raytracer plug-in for V-Ray. Atmospheric scattering was not used; these renders use $\rho$ as advected by the fluid for the emissive and absorbing factors for the volume tracer, with color dictated by a blackbody colormap.

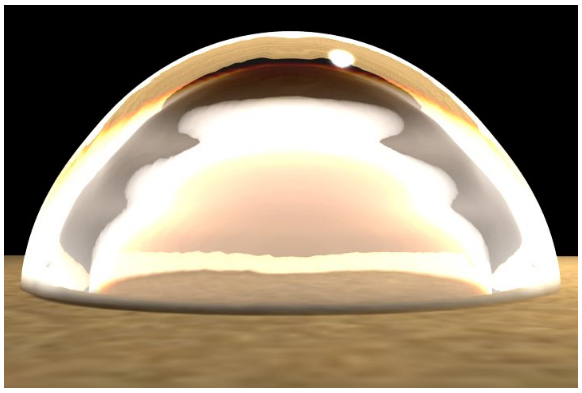


**Figure 6:** *The initial moments of the "Trinity test" — the first atomic bomb*

The two-dimensional demonstrations were rendered with our simple custom 2D plotting tool; those using a monochrome colormap demonstrate our method's preservation of sharp shock features through a *schlieren* plot — namely, we plot $\sqrt{|\nabla\rho|}$. The term schlieren refers to a particular type of image formed by the passage of light through inhomogeneous media that causes shadows to appear in areas of high inhomogeneity.

| Scene | resolution | sim. fps | avg. $\Delta t$ | sim. time |
|---|---|---|---|---|
| Blast chamber | $120\times80\times120$ | 1.56 | 1.4e-4 s | 16.25 min |
| Rigids w/ refl. | $60\times60\times100$ | 0.779 | 2.5e-4 s | 29.93 min |
| Tower (top) | $60\times80\times60$ | 1.14 | 7.2e-4 s | 30.21 min |
| Trinity | $200\times75\times200$ | 0.102 | 2.9e-5 s | 32.88 min |
| Tower (no top) | $60\times100\times60$ | 0.939 | 6.8e-4 s | 51.74 min |
| Mush. cloud | $120\times100\times120$ | 0.243 | 2.4e-2 s | 57.74 min |
| House | $100\times100\times100$ | 0.310 | 4.6e-5 s | 58.35 min |
| Projectile | $250\times100\times100$ | 0.191 | 1.0e-5 s | 191.5 min |

**Figure 7:** *Demonstrative timings of our method showing grid resolution, simulation frames per second, average simulation timestep, and the total computation time needed for the entire simulation run.*

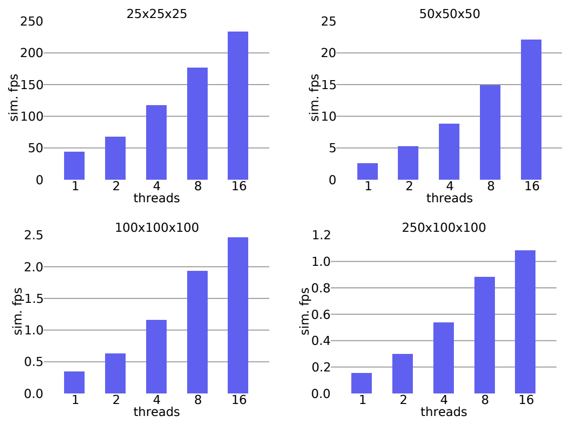


**Figure 8: Parallelization** *The performance of our algorithm increases with the number of threads. We measured performance as simulation frames per second with an OpenMP implementation benchmarked on a 16-core Intel Xeon machine.*

### 4.3. Timings

We present performance data in Fig. 7; these timings were collected on a 2GHz Core 2 laptop. Memory usage is linear in the number of grid cells — each demonstration fits within 500MB of memory. These timings are all for a single thread of computation; parallelization is discussed in Sec. 4.4.

Direct comparisons with previous works are difficult to produce because little or no timing information is available for these papers. Figure 2 in [YOH00] shows a 2D slice of a $101^3$ simulation of a shockwave interacting with a stationary wall; they reported a simulation time of 'overnight'. We reproduced this simulation with our method; for a $101^3$ grid, we recorded a total simulation time of *15 minutes*. Conservatively estimating that our hardware is nearly 7 times faster and that 'overnight' is about 10 hours, our method is at least 6 times faster than theirs at equivalent resolutions, and our simulation contains more visual detail.

To demonstrate the ability of our method to produce detailed results at coarse resolutions, we performed the same simulation on a $60^3$ grid; this took less than *2 minutes* (roughly 45x faster) and the generated results exhibit are more detailed than the results computed on a $101^3$ grid using [YOH00]. We have included these results in our supplementary video; the corresponding video from Yngve et al. can be found at `http://www.cs.berkeley.edu/b-cam/Papers/Yngve-2000-AE/Stuff/wall_pressure.mpeg`.

### 4.4. Parallelization

For the purpose of demonstrating that our algorithm is amenable to parallelization on shared-memory architectures, we have used OpenMP [ope05] to execute two essential computational kernels in separate work threads; the results are shown in Fig. 8. Due to dimensional splitting, the Riemann solves (step 1 in Sec. 3.3.4) are completely independent for each row in a computational sweeping plane along the simulation grid, as illustrated in Fig. 9; the update pass

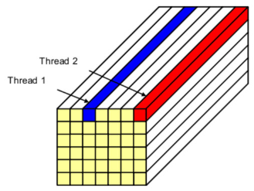


**Figure 9:** *Computation of Riemann solutions and solution updates done in a pass are divided among threads*

(step 3 in Sec. 3.3.4) can be similarly parallelized. We have performed benchmarks of our parallel implementation of simulations with increasing grid size on an Intel Xeon machine with sixteen cores, each running at 3GHz. The performance of our algorithm, here measured in simulation frames per second, scales well with the number of threads.

It should be noted that our attempt to parallelize the code is by no means a fully optimized implementation. A more optimized implementation takes into account such important concerns as data contention, cache sizes and parallel reductions. The key observation is that our approach can lead to extremely efficient implementations on parallel architectures with minimal effort.

## 5. Conclusion

We have presented a method for efficient simulations of supersonic flows in compressible, inviscid fluids that is based on the finite volume method. We have demonstrated the ability of our method to capture the behavior of shocks and to handle complex, bidirectional object-shock interactions stably. Additionally, we have demonstrated that our method scales well on modern architectures.

### 5.1. Limitations

Hyperbolic systems of equations (i.e. the compressible, inviscid Euler equations simulated here) are subject to the CFL condition as a requirement for convergence *and* stability. The unconditionally stable solvers popular for incompressible fluid dynamics are subject to the CFL condition for convergence, but not stability — indeed, the convention seems to take the CFL condition as a "guideline" and use CFL numbers upwards of 5.

Our technique performs well at simulating truly hyperbolic phenomena such as compressible, inviscid fluid dynamics, but cannot handle nearly incompressible phenomena (e.g. liquids) as efficiently as those simulations currently used in computer graphics. This fundamental limitation is due to the choice of equations — the actual propagation of acoustic waves so important to compressible fluids has a negligible effect on incompressible fluids.

### 5.2. Future work

There are a number of promising areas for future work. Many natural phenomena give rise to shocks — of particular interest to graphics are hydraulic jumps in the Saint-Venant (or shallow water) equations.

Additionally, the inherent parallelism of this approach provides ample opportunity for improved application to parallel hardware, and we intend to investigate this method on next-generation computing platforms.

**Acknowledgement:** This research is supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF Awards 0400134, 0429583, and 0636208, DARPA/RDECOM Contract N61339-04-C-0043, Intel, and Carolina Development.

## References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. In *ACM SIGGRAPH '07* (New York, NY, USA, 2007), ACM, p. 48.

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. In *ACM SIGGRAPH '07* (2007).

[BFMF06] BRIDSON R., FEDKIW R., MULLER-FISCHER M.: Fluid simulation: Siggraph 2006 course notes. In *ACM SIGGRAPH '06 Courses* (New York, NY, USA, 2006), ACM Press, pp. 1–87.

[ble] Blender 2.45. http://www.blender.org/.

[bul] Bullet Physics Library. http://www.bulletphysics.com/.

[CFL28] COURANT R., FRIEDRICHS K., LEWY H.: Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen 100*, 1 (1928), 32–74.

[CFL*07] CHENTANEZ N., FELDMAN B. E., LABELLE F., O'BRIEN J. F., SHEWCHUK J. R.: Liquid simulation on lattice-based tetrahedral meshes. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 219–228.

[CGFO06] CHENTANEZ N., GOKTEKIN T. G., FELDMAN B. E., O'BRIEN J. F.: Simultaneous coupling of fluids and deformable bodies. In *SCA '06: Proeedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2006), ACM Press/Addison-Wesley Publishing Co.

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. 23*, 3 (2004), 377–384.

[ETK*07] ELCOTT S., TONG Y., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1 (2007), 4.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *ACM SIGGRAPH '01* (New York, NY, USA, 2001), ACM Press, pp. 23–30.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483.

[FOA03] FELDMAN B. E., O'BRIEN J. F., ARIKAN O.: Animating suspended particle explosions. In *ACM SIGGRAPH '03* (New York, NY, USA, 2003), ACM, pp. 708–715.

[FOK05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M.: Animating gases with hybrid meshes. In *ACM SIGGRAPH '05* (New York, NY, USA, 2005), ACM Press, pp. 904–909.

[FSS03] FEDKIW R., SAPIRO G., SHU C.-W.: Shock capturing, level sets and PDE based methods in computer vision and image processing: A review on Osher's contribution. *J. Comput. Phys.*, 185 (2003), 309–341.

[GHD03] GENEVAUX O., HABIBI A., DISCHLER J.-M.: Simulating fluid-solid interaction. In *Proc. Graphics Interface '03* (2003).

[GSLF05] GUENELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. In *ACM SIGGRAPH '05* (New York, NY, USA, 2005), ACM Press, pp. 973–981.

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. In *ACM SIGGRAPH '06* (New York, NY, USA, 2006), ACM Press, pp. 820–825.

[Lev02] LEVEQUE R. J.: *Finite Volume Methods for Hyperbolic Problems*. Cambgridge University Press, New York, 2002.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH '04* (New York, NY, USA, 2004), ACM Press, pp. 457–462.

[LW60] LAX P., WENDROFF B.: Systems of conservation laws. *Comm. Pure Appl. Math.*, 13 (1960), 217–237.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159.

[MMA99] MAZARAK O., MARTINS C., AMANATIDES J.: Animating exploding objects. In *Proc. Graphics Interface '99* (Wellesley, MA, USA, 1999), AK Peters, pp. 211–218.

[NF99] NEFF M., FIUME F.: A visual model for blast waves and fracture. In *Proc. Graphics Interface '99* (Wellesley, MA, USA, 1999), AK Peters, pp. 193–202.

[ope05] OpenMP Version 2.5 Specification, May 2005. http://www.openmp.org/drupal/mp-documents/spec25.pdf.

[Roe81] ROE P.: Approximate Riemann solvers, parameter vectors, and difference schemes. *J Comput. Phys.*, 43 (1981), 357–372.

[SMML07] SEWALL J., MECKLENBURG P., MITRAN S., LIN M.: Fast fluid simulation using residual distribution schemes. In *Eurographics Workshop on Natural Phenomena 2007* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 47–54.

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH '05* (2005), pp. 910–914.

[Sta99] STAM J.: Stable fluids. In *Siggraph 1999, Computer Graphics Proceedings* (Los Angeles, 1999), Rockwood A., (Ed.), Addison Wesley Longman, pp. 121–128.

[vL77] VAN LEER B.: Towards the ultimate conservative difference scheme iv. *J. Comp. Phys.*, 22 (1977), 276–299.

[vra] V-Ray. http://www.chaosgroup.com/en/2/vray.html.

[WBOL07] WENDT J., BAXTER W., OGUZ I., LIN M.: Finite-volume flow simulations in arbitrary domains. *Graphical Models 69*, 1 (2007), 19–32.

[YOH00] YNGVE G. D., O'BRIEN J. F., HODGINS J. K.: Animating explosions. In *ACM SIGGRAPH '00* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 29–36.