Target-Driven Smoke Animation

Raanan Fattal[*] Dani Lischinski[†]

School of Computer Science and Engineering
The Hebrew University of Jerusalem

Abstract

In this paper we present a new method for efficiently controlling animated smoke. Given a sequence of target smoke states, our method generates a smoke simulation in which the smoke is driven towards each of these targets in turn, while exhibiting natural-looking interesting smoke-like behavior. This control is made possible by two new terms that we add to the standard flow equations: (i) a driving force term that causes the fluid to carry the smoke towards a particular target, and (ii) a smoke gathering term that prevents the smoke from diffusing too much. These terms are explicitly defined by the instantaneous state of the system at each simulation timestep. Thus, no expensive optimization is required, allowing complex smoke animations to be generated with very little additional cost compared to ordinary flow simulations.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: animation control, fluid dynamics, smoke simulation

1 Introduction

Animated natural phenomena, such as smoke, fire, and liquids, have become an indispensable component in contemporary computer generated animation, special effects, training simulators, and computer games. Currently, realistic and believable animations of these elements are most effectively achieved through physically-based numerical simulations of fluid dynamics. Efficient and stable methods for performing such simulations have been recently introduced to the computer graphics community [Stam 1999; Fedkiw et al. 2001; Foster and Fedkiw 2001; Enright et al. 2002; Nguyen et al. 2002]. However, providing animators with means for controlling such animations in an intuitive yet precise manner remains a challenging open problem. This paper addresses this problem by introducing a new tool for controlling animations of smoke.

To illustrate just how challenging smoke animation control can be, consider the shot from the recent feature film *The Fellowship of the Ring*, where the smoke that Gandalf blows forms a beautiful galleon that sails right through Bilbo's expanding smoke ring. It would have been extremely difficult, if not impossible, for an animator to achieve such detailed and precise smoke formations and motion by manually adjusting wind fields and other smoke simulation parameters.

In a recent pioneering work Treuille *et al.* [2003] introduced a method for keyframe control of smoke simulations. They employ a shooting technique to solve for the optimal wind forces that would cause a smoke simulation to approximate a set of user-specified keyframes as closely as possible. While this method has produced some very impressive results, it is highly computationally intensive, because it solves a very difficult multi-dimensional non-linear optimization problem. We discuss this method and its drawbacks in more detail in section 2.1.

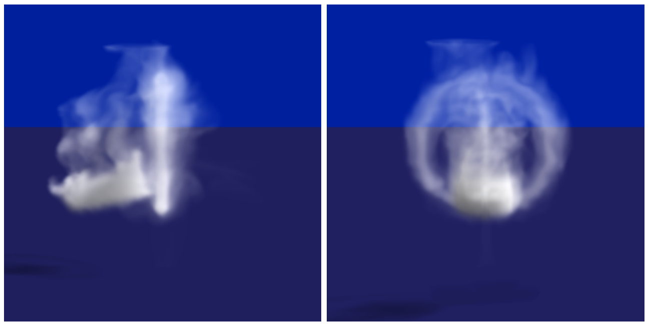
Figure 1: A smoke galleon sails through a ring of smoke (side and front views).

Inspired by the work of Treuille *et al.* [2003], we also focus on smoke animation control, but explore a different control paradigm: *target-driven* smoke animation. Rather than attempting to optimally approximate a set of keyframes, we control the simulation by a sequence of target smoke states, each serving in turn as an *attractor* for the simulated smoke. This is achieved by augmenting the standard fluid dynamics equations with closed-form terms designed specifically to drive the simulation from any given state towards a user-specified smoke density target. Our control paradigm does not guarantee that each target is approximated in an optimal manner before switching to the next one, but it eliminates the need for expensive non-linear optimization. Thus, complex and interesting smoke animations, such as the one shown in Figure 1, are generated with very little additional cost compared to an ordinary smoke simulation.

Specifically, our main contributions are: (i) a new *driving force* term in the momentum equation, designed to induce the fluid flow that will carry the current smoke density towards a user-specified smoke density target; (ii) a new *gathering* term in the smoke advection equation, designed to counteract diffusion of smoke due to numerical dissipation, thereby improving the ability of the simulation to match arbitrary targets; (iii) we provide animators with the ability to independently control several smoke fields sharing a fluid.

The remainder of this paper is structured as follows. In the next section we briefly survey relevant previous work, and review the standard equations commonly used for simulating fluid flow. In section 3 we derive our modified flow equations. Section 4 describes the implementation of our simulator. We present some of the results we were able to achieve with our technique in section 5. Section 6 concludes the paper.

[*]e-mail: raananf@cs.huji.ac.il
[†]e-mail: danix@cs.huji.ac.il

2 Background

2.1 Previous work

Many researchers in the field of computer graphics have used physically-based models and computational fluid dynamics (CFD) to simulate fluid flows. We will not attempt to provide a complete survey of such methods; the interested reader can find good surveys in the previous work sections of [Stam 1999], [Fedkiw et al. 2001], and [Treuille et al. 2003]. Instead, we focus on the different mechanisms for controlling fluid flows.

Foster and Metaxas [1997] describe a mechanism referred to as "embedded controllers" via which an animator may introduce various effects into fluid animations by adapting boundary and fluid properties, and pressure and velocity fields. Thus, animators need not understand the fluid dynamics equations, nor the low-level details of the solver. However, they must understand the dynamics of the effect they are interested in achieving.

Witting [1999] describes the CFD tool that was used at DreamWorks Feature Animation to produce smoke and fluid effects for the feature film *The Prince of Egypt*. With this tool animators drive fluid simulations by using images and animated sequences to specify inputs such as initial temperature fields, heat sources, and forcing functions. Another production tool that provides animators both artistic and behavioral control for animation of flames is described by Lamorlette and Foster [2002]. However, neither of these tools support direct control over the desired results.

As already mentioned earlier, the most relevant previous work, which inspired our own, is that of Treuille *et al.* [2003]. In their approach an animator controls a smoke simulation by specifying smoke density keyframes. Continuous quasi-Newton optimization is then used to solve for the control forces that would cause the simulated smoke to approximate the keyframes while minimizing the amount of force. The control force field is defined by a collection of parametric wind and vortex forces, and the optimization process searches for the appropriate parameter values.

While this approach has produced some very impressive results, it suffers from several drawbacks. First, the optimization framework is very expensive, as it requires multiple evaluations of the objective function and of its gradient. Each evaluation performs an entire simulation and, in addition to computing the velocity and the smoke density, their derivatives with respect to each of the control parameters are computed as well. Furthermore, the dimensionality of the optimization problem grows with the length of the simulation, so it is necessary to split the original problem into several smaller optimization problems, organized in two overlapping schedules, and to alternate between them. The dimensionality and cost concerns also dictate using a small number of control forces and a relatively coarse grid, making it difficult to specify and match highly detailed keyframes.

In contrast, the approach described in this paper does not involve solving an optimization problem, and a target-driven smoke animation may be obtained at nearly the same cost as a single ordinary simulation. Our approach does not require keeping track of derivatives, and thus we are free to use non-differentiable numerical schemes, such as flux limiter based hyperbolic solvers [Leveque 2002], which improve the accuracy of our simulations.

On the other hand, it should be pointed out that our target-driven approach does not provide direct control over the quality with which targets are matched. Instead, the animator is provided with several intuitive parameters for controlling the rate at which smoke evolves, as well as other characteristics of the simulation.

Also, we do not attempt to minimize the forces involved in the process, and the resulting simulation is not always physically realizable, because we employ a non-physical gathering term. However, our results demonstrate that interesting and complex smoke animations may be obtained without these additional constraints.

2.2 Problem statement

Let us denote by $\rho = \rho(\mathbf{x}, t)$ the time-dependent scalar field that specifies the density of smoke at location $\mathbf{x}$ and time t. Given an initial smoke density, $\rho_0 = \rho(\mathbf{x}, 0)$, and a sequence of target densities $\rho_i^* = \rho^*(\mathbf{x}, t_i)$, our goal is to produce an animation in which the smoke is driven towards each of these targets in turn, while maintaining smoke-like dynamics and appearance. More specifically, during each time interval (t_{i-1}, t_i) the smoke should evolve towards the target ρ_i^*. Note that unlike in traditional keyframe animation our problem statement does not require the i-th target to be matched precisely at time t_i, but rather that at time t_i the i-th target ceases to attract the smoke and ρ_{i+1}^* becomes the attractor instead.

2.3 The equations of flow

Realistic smoke animations are typically generated by numerically approximating either the Navier-Stokes or the Euler equations.[1] These equations govern the mechanics of a medium fluid (thin air) in which smoke is present. Denoting the fluid velocity vector field by $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ and its temporal derivative by $\mathbf{u}_t$, the inviscid Euler equations for incompressible flow are:

$$\begin{aligned} \mathbf{u}_t &= -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \mathbf{f} && (1) \\ \nabla \cdot \mathbf{u} &= 0 && (2) \end{aligned}$$

where $p = p(\mathbf{x}, t)$ is the hydrostatic pressure and $\mathbf{f}$ accounts for external forces affecting fluid flow, such as gravity and buoyancy. Equation 1 is the balance of momentum equation (Newton's second law) for a fluid with unit density. Equation 2 is the *continuity equation*, which enforces conservation of mass. An additional equation describes the transport of smoke by the fluid flow (advection):

$$\rho_t = -\mathbf{u} \cdot \nabla \rho \qquad (3)$$

Here ρ_t is the temporal derivative of the smoke density field ρ.

The external forces term $\mathbf{f}$ in equation 1 provides an important means of control over the smoke simulation. For example, Stam [1999] used this term to allow a user to control the flow by dragging a mouse. Treuille *et al.* [2003] include in this term a set of parameterized wind and vortex forces. Their optimization process then searches for parameter values that cause the simulation to match the specified keyframes. Thus, in their approach $\mathbf{f}$ implicitly depends on the initial smoke density ρ_0 and the target density ρ^*. Our approach also utilizes the force term in order to drive the simulation from a given initial state to a target state, but this is achieved by introducing a driving force term that depends only on the instantaneous state of the system, rather than being defined by some global optimization condition; it is an explicit yet simple function $\mathbf{f} = \mathbf{F}(\rho, \rho^*)$.

3 Modified Equations of Flow

Our approach entails two modifications to the fluid flow equations presented above. First, as already mentioned, we set the external force term $\mathbf{f}$ in equation 1 to $\mathbf{F}(\rho, \rho^*)$ and add a momentum attenuation term $-\nu_d\, \mathbf{u}$:

$$\mathbf{u}_t = -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \nu_f\, \mathbf{F}(\rho, \rho^*) - \nu_d\, \mathbf{u} \qquad (4)$$

We shall refer to $\mathbf{F}$ as the *driving force term*, since it is designed to exert forces on the fluid in such a manner that the resulting flow will carry the smoke to the prespecified target density ρ^*. This term is

[1]The Navier-Stokes equations contain a viscosity term, not present in the inviscid Euler equations.

derived in section 3.1. The momentum attenuation term is added as a means for limiting the accumulation of momentum in the system.

Second, we add a *smoke gathering term* $\mathbf{G}(\rho,\rho^*)$ to equation 3:

$$\rho_t \quad = \quad -\mathbf{u}\cdot\nabla\rho + \nu_g\,\mathbf{G}(\rho,\rho^*) \tag{5}$$

The purpose of this term is to counteract numerical dissipation that causes smoke to diffuse. It enables our simulations to closely approximate a target density field, even if it is "sharper" than the initial density field. We derive the gathering term in section 3.2.

These new terms are controlled by the non-negative parameters ν_f, ν_d, and ν_g, whose effect is discussed in section 3.4.

3.1 The driving force

In this section we develop an appropriate expression for the driving force term $\mathbf{F}$. Recall that the purpose of this force, at any given time t, is to cause the fluid to advect the current smoke density $\rho(\mathbf{x},t)$ towards the next target $\rho^*(\mathbf{x})$. Furthermore, we must also take care to define $\mathbf{F}$ in a manner that would enable the fluid to reach a rest state, $\mathbf{u}=\mathbf{u}_t=0$, once the target has been reached. Failure to enforce this requirement will cause undesirable fluctuations even in regions where $\rho=\rho^*$.

The rest state requirement may be satisfied by ensuring that the amount of momentum generated by $\mathbf{F}$ decreases as ρ approaches ρ^*, allowing the momentum attenuation term to bring the system to $\mathbf{u}=\mathbf{u}_t=0$ when $\rho=\rho^*$. In this state, equation 4 reduces to

$$\mathbf{F}(\rho^*,\rho^*) = \nabla p.$$

This means that rather than forcing $\mathbf{F}$ to vanish, we can simply allow the hydrostatic pressure to cancel it out. Thus, it suffices to ensure that $\mathbf{F}(\rho^*,\rho^*)$ is a gradient of some potential field, rather than an arbitrary vector field.

Note that the gradient of the target density $\nabla\rho^*$ always points uphill towards higher concentrations of ρ^*. This means that the gradient is the desired direction of flow, and to induce it we align the driving force at any point $\mathbf{x}$ with $\nabla\rho^*(\mathbf{x})$:

$$\mathbf{F}(\rho,\rho^*) \propto \nabla\rho^*$$

In practice, however, ρ^* may be constant in some regions of the domain, causing $\nabla\rho^*=0$ there. In order to avoid this problem we use a blurred version of ρ^*, which we denote $\tilde{\rho}^*$. In order to ensure that $\nabla\tilde{\rho}^*\neq 0$ everywhere, the blurring filter must have sufficiently large support. The filter should also fall off rapidly in order to avoid unnecessarily smoothing the directions of the gradients. In our implementation we obtain $\tilde{\rho}^*$ by convolving ρ^* with a Gaussian kernel given by

$$g(\mathbf{x}) = e^{-\mathbf{x}^\top\mathbf{x}/\sigma^2} \tag{6}$$

with a small value for σ. Although the gradient $\nabla\tilde{\rho}^*$ is non-zero everywhere, its magnitude is very small away from regions where the target is defined, because of the rapid falloff of the blurring filter. Therefore, we use a "normalized" gradient instead:

$$\mathbf{F}(\rho,\rho^*) \propto \frac{\nabla\tilde{\rho}^*}{\tilde{\rho}^*}.$$

Note that we are guaranteed that $\tilde{\rho}^*>0$, since it is obtained by convolving the non-negative function ρ^* with a Gaussian. As we shall see shortly, this form of normalization enables the driving force to comply with the rest state requirement.

In order to avoid applying forces unnecessarily, the magnitude of the driving force should be roughly proportional to the actual smoke density ρ. This could have been achieved by setting

$$\mathbf{F}(\rho,\rho^*) = \rho\,\frac{\nabla\tilde{\rho}^*}{\tilde{\rho}^*},$$

however, this expression does not comply with the rest state requirement. In order to fix this it suffices to replace ρ with its blurred version $\tilde{\rho}$:

$$\boxed{\mathbf{F}(\rho,\rho^*) = \tilde{\rho}\,\frac{\nabla\tilde{\rho}^*}{\tilde{\rho}^*}.} \tag{7}$$

Note that for $\rho=\rho^*$ this definition gives us

$$\mathbf{F}(\rho^*,\rho^*) = \tilde{\rho}^*\,\frac{\nabla\tilde{\rho}^*}{\tilde{\rho}^*} = \nabla\tilde{\rho}^*,$$

and thus $\mathbf{F}(\rho^*,\rho^*)$ is indeed a gradient of a potential field.

3.2 Smoke gathering

In general, it is not possible to match a given target density field solely by advection. Consider, for example, a case where the target field ρ^* contains higher values and higher gradients than those present in the initial field ρ_0. In the course of a simulation these initial values and gradients only decrease because of numerical dissipation, and therefore will never achieve the target values. Even if we were to somehow avoid numerical dissipation, it would still be impossible to match such a target: from the mathematical standpoint, due to incompressibility of the fluid, any advection applied to ρ_0 is merely a spatial remapping of this density distribution, incapable of generating any new density values. Tackling this issue by introducing an inverse diffusion operator, such as $-\alpha\nabla^2\rho$, is physically unstable [Press et al. 1992]. Instead, we introduce an alternative mechanism, which we refer to as *smoke gathering*. This mechanism is enforced by the $\mathbf{G}(\rho,\rho^*)$ term in equation 5.

Let us assume that the total mass of the simulated smoke (the integral of ρ over the entire domain) is equal to that of the target smoke.[2] Now consider the time varying "density error" field $e(\mathbf{x},t)=\rho(\mathbf{x},t)-\rho^*(\mathbf{x})$. Obviously, any large differences in the values and/or gradients between ρ and ρ^* give rise to high values and high gradients in e. The essence of our idea is to apply diffusion to e gradually smoothing it as time goes by. This is a stable process that will eventually make e constant across the entire domain, which implies $\rho=\rho^*$, because of mass conservation.

Formally, we would like the density error e to satisfy the following diffusion equation:

$$e_t = \nabla^2 e \tag{8}$$

with the von Neumann boundary conditions

$$\frac{\partial e}{\partial \mathbf{n}} = 0,$$

which essentially state that the equation must be resolved inside the domain, as no flux of e across the boundary is allowed. As time advances, e becomes zero across the domain, which implies that

$$e=\rho-\rho^*=0 \quad \text{and} \quad \nabla e = \nabla\rho-\nabla\rho^*=0.$$

Thus, diffusing the error enables ρ to evolve exactly into ρ^*, given enough time.

Substituting the definition of e into equation 8 and noting that ρ^* is constant with respect to time, we obtain the following equation in ρ:

$$\rho_t = \nabla^2(\rho-\rho^*). \tag{9}$$

Thus, in order to achieve diffusion of the error, we could simply set $\mathbf{G}(\rho,\rho^*)=\nabla^2(\rho-\rho^*)$ in equation 5. In practice, however, we'd

[2] We can always scale the source or the target to make sure that the two masses are equal to each other.

like the gathering effect to occur only in the close vicinity of the target ρ^*, otherwise we end up simply diffusing ρ:

$$\rho_t = \nabla^2(\rho - \rho^*) \approx \nabla^2\rho.$$

In addition, gathering should only be allowed to occur where some smoke ρ is actually present, otherwise we might obtain negative values of ρ. To account for these requirements we rewrite equation 9 as

$$\rho_t = \nabla \cdot \nabla(\rho - \rho^*),$$

where $\nabla(\rho - \rho^*)$ is the error flux, and modulate the error flux by both $\tilde{\rho}^*$ and ρ. Thus, our gathering term becomes

$$\boxed{\mathbf{G}(\rho,\rho^*) = \nabla \cdot [\rho\tilde{\rho}^*\nabla(\rho - \rho^*)].} \tag{10}$$

Note that in this modified form the gathering term is still a conservation law, so ρ and hence e are conserved.

Figure 2 shows frames from a simple animation sequence computed with and without the gathering term (rows B and A, respectively). Note that without the gathering term the simulation fails to converge to its target. By enabling the gathering only in the vicinity of the target no diffusion is introduced in the early stages of the simulation.

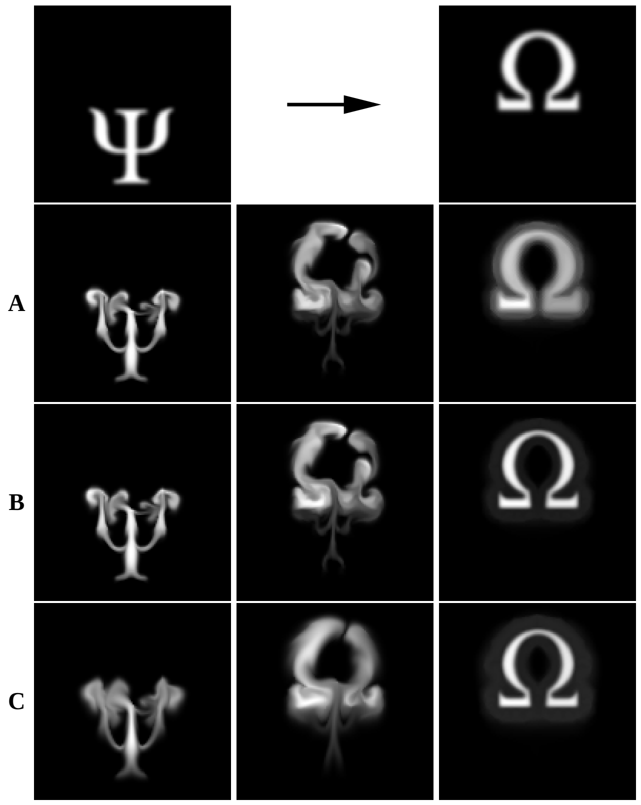

Figure 2: The top row shows the initial density ρ_0 on the left and the target density ρ^* on the right. Each of the next three rows shows three frames from a resulting simulation. Row **A** was generated with the gathering term disabled, and the simulation fails to converge to the target. Turning gathering on enables the simulation to converge (row **B**). Row **C** was generated using a first order hyperbolic solver. The resulting frames are blurrier and contain less detail than those in row **B**, which were generated using a second order solver. The full sequences are available on the ACM SIGGRAPH 2004 Full Conference DVD-ROM.

3.3 Multiple field flow

Our driving force term attracts any smoke present in the system towards the nearest "peak" of the target density. To provide the animator with more precise control over the affinity between the smoke and the target it is sometimes useful to split the smoke in the system to a number of independent density fields, each with its own set of initial and target states. Although each smoke field is controlled independently, there is an implicit interaction between them, as they are advected by, and exert driving forces on, the same fluid. Given n smoke fields $\rho^1,\ldots,\rho^n$ and their corresponding targets $\rho^{*,1},\ldots,\rho^{*,n}$ the only change in equation 4 is that the driving force is now the sum of the n driving forces $\mathbf{F}(\rho^i,\rho^{*,i})$ and we now have n advection equations for the smoke fields, instead of one:

$$\rho^i_t \quad = \quad -\mathbf{u}\cdot\nabla\rho^i + v_g\,\mathbf{G}(\rho^i,\rho^{*,i})$$

Note, however, that there is still only one velocity field $\mathbf{u}$.

3.4 Control parameters

The simulations produced by our methods are controlled via four parameters: the force smoothing parameter σ (eq. 6), the force coefficient v_f, the momentum attenuation coefficient v_d (eq. 4), and the gathering coefficient v_g (eq. 10):

σ – Increasing the value of this parameter results in a smoother driving force, inducing more moderate flows. Smoke is advected towards its target in a more organized manner, with less distortion and splitting along the way. On the other hand, increasing σ makes it more difficult for the flow to cause ρ to form the finer features of ρ^*. Decreasing σ creates more turbulent flows and a more chaotic smoke evolution.

v_f – This parameter allows the animator to boost or weaken the driving force, thus affecting the speed at which the simulation progresses towards the current target state.

v_d – This parameter determines the rate of momentum attenuation. Increasing it results in a more controlled, viscous-like flow, making it less likely for smoke to gain too much momentum and overshoot its target.

v_g – This parameter determines the rate at which ρ is gathered towards ρ^*. Increasing it reduces the amount of "stray" smoke in the simulation. On the other hand, setting the gathering rate too high makes the convergence of ρ to ρ^* appear less natural, as discussed in section 5.1.

Animations demonstrating the effect of each of these parameters are available on the DVD-ROM.

4 Numerical simulation

Given an initial velocity field $\mathbf{u}^0$, initial smoke density ρ_0, and a target smoke density ρ^* we simulate the fluid flow and the resulting motion of smoke by numerically solving equations 2, 4, and 5 over a series of discrete time steps.

Equations 4 and 5 express the rate of change of $\mathbf{u}$ and ρ as a sum of terms, each representing a different contribution. Such equations are often solved by the method of *fractional steps*, also known as *operator splitting* [Press et al. 1992]. The same approach is used by Stam [1999] and by Fedkiw *et al.* [Fedkiw et al. 2001]. The main idea behind this method is to solve the original equation by integrating a succession of simpler equations, each with a single term on its right hand side. Each of the simpler equations can be treated more effectively by choosing the most appropriate method for integrating it. Specifically, we perform the following sequence of fractional steps in each time step:

1. Apply driving forces: $\mathbf{u}_t = v_f\, \mathbf{F}(\rho, \rho^*)$
2. Attenuate momentum: $\mathbf{u}_t = -v_d\, \mathbf{u}$
3. Advect momentum: $\mathbf{u}_t = -\mathbf{u} \cdot \nabla \mathbf{u}$
4. Project: solve for the pressure field p given by the Poisson equation $\nabla^2 p = \nabla \cdot \mathbf{u}$ and subtract ∇p from $\mathbf{u}$.
5. Advect smoke: $\rho_t = -\mathbf{u} \cdot \nabla \rho$
6. Gather smoke: $\rho_t = v_g\, \mathbf{G}(\rho, \rho^*)$.

In our implementation, steps 3, 5 and 6 are restricted by a CFL condition resulting from the explicit hyperbolic scheme used at these steps (described in appendix A). Formally, step 2 imposes another restriction on the maximal time step to be less than $1/v_d$. In practice, this is not a limitation as the typical values of v_d are quite small.

In the remainder of this section we describe our specific implementation of these steps. We employ standard state-of-the-art CFD techniques, described here for completeness. Thus, readers not interested in implementation details may safely skip the remainder of this section.

4.1 The computational grid

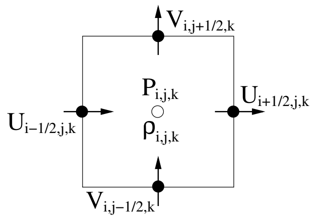

Figure 3: Variables on the computational grid (in 2D).

We use a regular 3D grid to solve our equations numerically using finite differences. We use a staggered arrangement of the different variables [Harlow and Welch 1965]: the velocity variables are defined at the centers of the cell faces, while the pressure and density variables are defined at the center of each cell (see Figure 3). Such an arrangement avoids the infamous checkerboard instability [Trottenberg et al. 2001], and it has also been used in previous fluid simulators in computer graphics [Foster and Metaxas 1996; Fedkiw et al. 2001]. We use the half-way index notation to distinguish between the different variable locations: for example, the variables for velocity components along the x, y, and z axes are denoted $U_{i\pm\frac{1}{2},j,k}$, $V_{i,j\pm\frac{1}{2},k}$, and $W_{i,j,k\pm\frac{1}{2}}$, respectively, while the density variables are denoted $\rho_{i,j,k}$. Thus, $U_{i+\frac{1}{2},j,k}$ is located at $((i+1/2)\Delta x, j\Delta y, k\Delta z)$, where Δx, Δy, and Δz are the the cell spacings along the x, y, and z axes, respectively. When necessary, we denote the number of the fractional step using a parenthesized superscript.

4.2 Discrete operators

For compactness and readability we introduce notation for several discrete operators. First derivatives are approximated halfway between the positions of the corresponding variable. For example, the first derivative of the velocity along the x axis is approximated as

$$\mathscr{D}_x(U)_{i,j,k} = \frac{1}{\Delta x}(U_{i+\frac{1}{2},j,k} - U_{i-\frac{1}{2},j,k})$$

and the first derivative of the density along the y axis is

$$\mathscr{D}_y(\rho)_{i,j+\frac{1}{2},k} = \frac{1}{\Delta y}(\rho_{i,j+1,k} - \rho_{i,j,k}).$$

Second derivatives are approximated at the positions where the variables are defined. For example,

$$\mathscr{D}_{zz}(W)_{i,j,k+\frac{1}{2}} = \frac{1}{\Delta z^2}(W_{i,j,k+\frac{3}{2}} - 2W_{i,j,k+\frac{1}{2}} + W_{i,j,k-\frac{1}{2}})$$

is the second derivative of the velocity along the z axis.

Occasionally, we require the value of a field halfway between the locations where its variables are defined. In such cases, we approximate the value by using simple averaging. For example,

$$\mathscr{A}(V)_{i,j,k} = \frac{1}{2}(V_{i,j+\frac{1}{2},k} + V_{i,j-\frac{1}{2},k})$$

is the velocity along the y axis at the center of a cell.

4.3 Applying the driving force

Components of the driving force must be evaluated for each updated velocity variable at its corresponding location. We denote these components $F^u_{i\pm\frac{1}{2},j,k}$, $F^v_{i,j\pm\frac{1}{2},k}$, and $F^w_{i,j,k\pm\frac{1}{2}}$. Each of these components is evaluated by approximating equation 7 at the corresponding grid point, using a discrete derivative operator and averaging:

$$F^u_{i+\frac{1}{2},j,k} = \mathscr{A}(\tilde{\rho})_{i+\frac{1}{2},j,k} \frac{\mathscr{D}_x(\tilde{\rho}^*)_{i+\frac{1}{2},j,k}}{\mathscr{A}(\tilde{\rho}^*)_{i+\frac{1}{2},j,k}}$$

and similarly for F^v and F^w. The velocity variables are then updated as follows:

$$U^{(1)}_{i+\frac{1}{2},j,k} = U^{(0)}_{i+\frac{1}{2},j,k} + \Delta t\; v_f\; F^u_{i+\frac{1}{2},j,k}$$

Note that when $\rho = \rho^*$, the driving force is "projected out" by the projection step in the *discrete sense*, allowing the discrete velocity variables to be at rest. This is due to the fact that in this case the discrete force components reduce to

$$F^u_{i+\frac{1}{2},j,k} = \mathscr{D}_x(\tilde{\rho}^*)_{i+\frac{1}{2},j,k}$$

that is, the gradient of a discrete potential function, which is exactly what is being eliminated by the projection step.

4.4 Advection and Gathering terms

The advection term appearing in fractional steps 2 and 5 can be written in conservation form

$$q_t \quad = \quad -\mathbf{u} \cdot \nabla q \quad = \quad -(uq)_x - (vq)_y - (wq)_z$$

where q stands for either $\mathbf{u}$ or ρ. We solve this equation as three separate 1D hyperbolic equations

$$\begin{aligned} q_t &= -(uq)_x \\ q_t &= -(vq)_y \\ q_t &= -(wq)_z \end{aligned}$$

using the high resolution hyperbolic solver described in Appendix A. This is a second-order numerical scheme that is considerably less prone to numerical dissipation compared to first-order schemes. Thus, for a given grid resolution more turbulent flows are obtained, and the fine features of the evolving smoke are better preserved.

These advantages are demonstrated in Figure 2, where row C shows the result of first order upwinding, compared to the second order scheme in row B.

The gathering term defined by equation 10 is already given in conservation form, and is also solved by the same solver, independently for each axis:

$$\begin{aligned}\rho_t &= [\rho\tilde{\rho}^*(\rho-\rho^*)_x]_x\\ \rho_t &= [\rho\tilde{\rho}^*(\rho-\rho^*)_y]_y\\ \rho_t &= [\rho\tilde{\rho}^*(\rho-\rho^*)_z]_z\end{aligned}$$

4.5 Projection

The discrete velocity field $(U,V,W)^{(3)}$ after fractional step 3, is not divergence free. To impose equation 2 discretely, we use Chorin's projection technique [1967]. We recover a divergence free field $(U,V,W)^{(4)}$ in the discrete sense, i.e.

$$(\mathscr{D}_x(U^{(4)})+\mathscr{D}_y(V)^{(4)}+\mathscr{D}_z(W)^{(4)})_{i,j,k}=0,$$

by subtracting the gradient of the pressure field P:

$$\begin{aligned}U^{(4)}_{i+\frac{1}{2},j,k} &= U^{(3)}_{i+\frac{1}{2},j,k}-\mathscr{D}_x(P)_{i+\frac{1}{2},j,k}\\ V^{(4)}_{i,j+\frac{1}{2},k} &= V^{(3)}_{i,j+\frac{1}{2},k}-\mathscr{D}_y(P)_{i,j+\frac{1}{2},k}\\ W^{(4)}_{i,j,k+\frac{1}{2}} &= W^{(3)}_{i,j,k+\frac{1}{2}}-\mathscr{D}_z(P)_{i,j,k+\frac{1}{2}}\end{aligned}$$

Combining the above equations, we get the following set of equations for the pressure variables $P_{i,j,k}$

$$\begin{aligned}(\mathscr{D}_{xx}(P)+\mathscr{D}_{yy}(P)+\mathscr{D}_{zz}(P))_{i,j,k} &= \\ &(\mathscr{D}_x(U^{(3)})+\mathscr{D}_y(V^{(3)})+\mathscr{D}_z(W^{(3)}))_{i,j,k},\end{aligned} \tag{11}$$

which is a discretization of the Poisson equation $\nabla^2 p = \nabla\cdot\mathbf{u}$. This is a large sparse set of linear equations, which we solve using a standard multigrid solver [Trottenberg et al. 2001].

The values of the velocity variables on the boundary are prescribed by the boundary conditions, so there we have $(U,V,W)^{(4)} = (U,V,W)^{(3)}$. Consequently, for the left boundary cells, equation 11 reads

$$\begin{aligned}&\frac{1}{\Delta x^2}\left(P_{1,j,k}-P_{0,j,k}\right)+\mathscr{D}_{yy}(P)_{0,j,k}+\mathscr{D}_{zz}(P)_{0,j,k} =\\ &\frac{1}{\Delta x}\left(U^{(3)}_{\frac{1}{2},j,k}-U^{(3)}_{-\frac{1}{2},j,k}\right)+\mathscr{D}_y(V^{(3)})_{0,j,k}+\mathscr{D}_z(W^{(3)})_{0,j,k}.\end{aligned}$$

This is also the case for all the other boundaries.

5 Results and Discussion

We have implemented our method in 2D and in 3D using the C++ programming language. All of the timings reported below were measured on a 2.4GHz Pentium IV with 1GB of RAM running Linux. For 2D simulations on a 256^2 grid a single time step takes 0.25 seconds. For 3D simulations on a 128^3 grid each time step takes 10.6 seconds. This is only about 15 percent slower than the computation time for an ordinary smoke simulation (without our driving force and gathering terms) using the same implementation.

In all of the examples shown in this section we used between 5 and 15 adaptive timesteps for each animation frame, so each second of a 20fps animation took about 50 seconds of simulation time in 2D, and about 35 minutes in 3D. This is faster by at least two orders of magnitude than the running times in [Treuille et al. 2003], where they report that it took 2–5 hours to optimize a 50×50 2D simulation.

We experimented with two different kinds of smoke animations. In the first kind we use target states that are very different from each other, in essence producing a morphing sequence between pairs of states. Examples are shown in Figures 4 and 5.

Figure 4 shows several frames from a transforming 2D logo sequence. Despite the very different and complex target states our method manages to approximate the targets quite well, while maintaining fluid-like behavior. Note that the entire simulation was performed on a single grid without any spatial or temporal splitting, enabling interesting global interactions: the smoke starts out as a single body, then splits to form separate letters, which later join again to form the final logo.

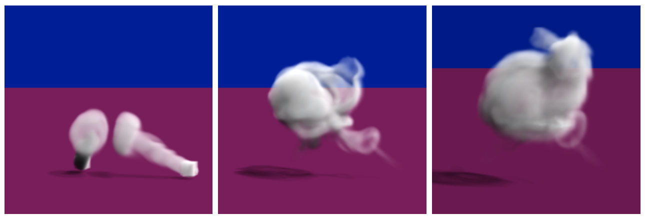
Figure 5: The Stanford bunny is formed by smoke emitted from two sources.

Figure 5 is an example where rather than morphing between two targets we specify a single target and two smoke sources that generate smoke during the first half of the animation. The driving force is very effective at directing the two turbulent smoke streams from the sources to accurately form the shape of the Stanford bunny.

Another kind of smoke animations are those defined by a dense sequence of similar targets. The idea is that we can voxelize an animated 3D object and use the resulting volumes to drive a smoke animation. Frames from two such animations are shown in Figure 6. Here the overall motion is dictated by the targets, and the characters are still quite recognizable, but they exhibit interesting smoke-like dynamics. The gathering term is instrumental here, as it enables the smoke to track the moving character for as long as needed.

As discussed in section 3.3, our system allows animators to control several smoke fields independently. Figure 7 demonstrates this for two smoke fields, each visualized using a different color. Here the use of two fields allows us to specify that the right half of the

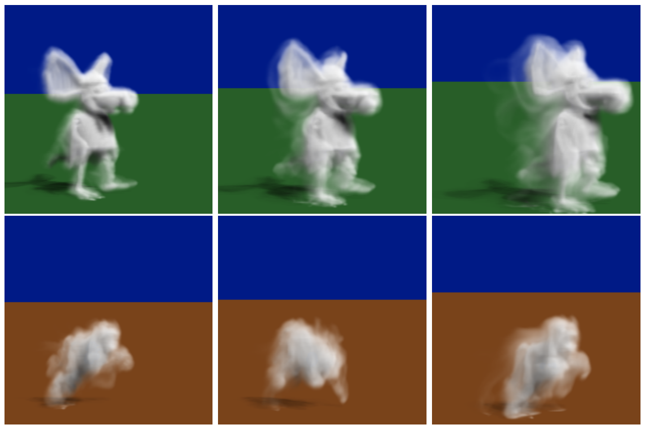
Figure 6: Top row: a walking mouse. Bottom row: a leaping tiger.

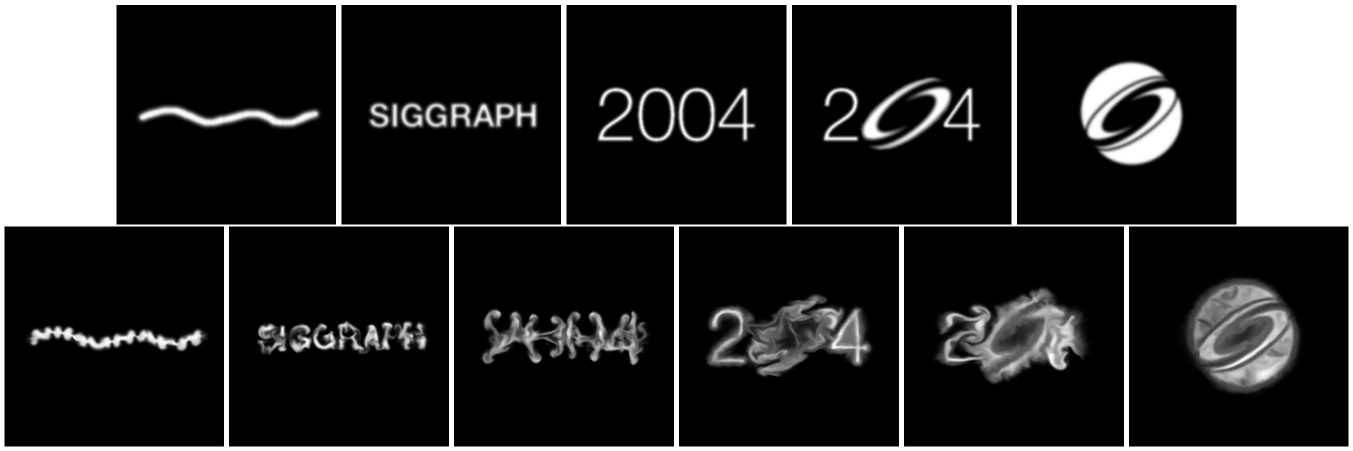

Figure 4: Fluid logo. The five targets are shown in the top row, and six frames from the resulting animation are shown in the bottom row.

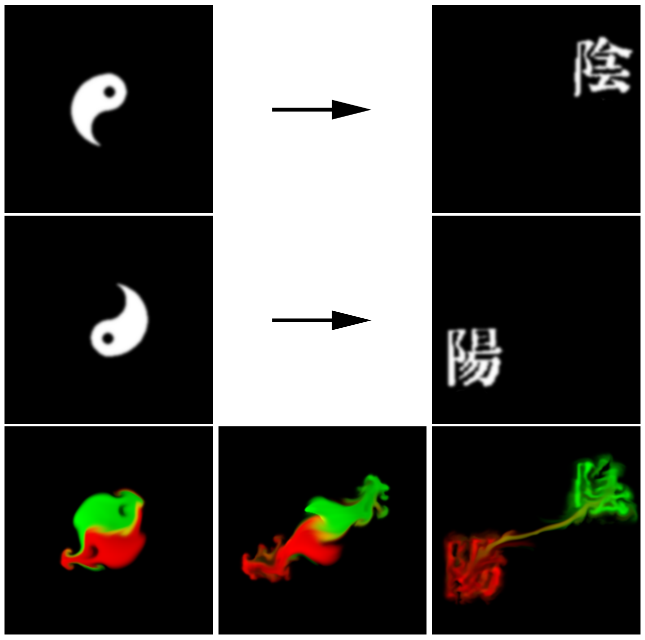

Figure 7: Yin-Yang: control of two smoke fields.

circle should flow towards the lower left corner, while the left half should flow to the upper right, which could not have been achieved with a single smoke field.

Finally, Figure 8 shows a 3D animation which utilizes a combination of all of the elements described above. In one smoke field a sphere evolves into a ring of smoke, while in another field the smoke tracks the shape of a galleon sailing through the smoke ring. Note that the two fields interact with each other via their shared medium fluid.

5.1 Limitations

Smoke diffusion is unavoidable in numerical simulations. As pointed out in section 3.2, it presents a serious problem in the context of controlled smoke animations. The gathering term we have introduced does provide a partial remedy to this problem, as it enables the simulation to converge to an arbitrary target. However, the resulting smoke transition does not always come across as a natural evolution of smoke, since it is based on a diffusion process and not on kinetics. For example, the animation might sometimes look like the target is simply emerging from an amorphous static cloud of smoke. Thus, the gathering mechanism should be employed to the least extent necessary, leaving the task of forming the target, as much as possible, to the advection induced by the driving force.

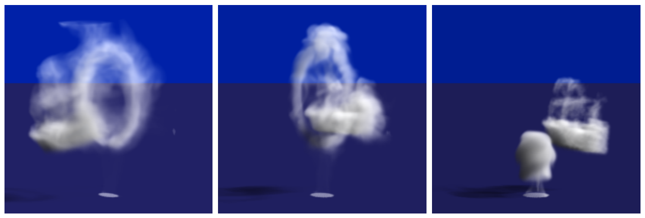

Figure 8: A galleon of smoke sailing through a smoke ring.

As pointed out earlier, the target-driven paradigm does not guarantee optimal approximation of the targets, and the animator cannot directly control how well a particular target is approximated at a specific instant in time. However, the animator may control the simulation progress in a less direct manner via the v_f parameter. Mechanisms for more precise time and accuracy control would make a good topic for further research.

Finally, just as in traditional keyframe animation, skill and experience are still required in order to choose the right target states and the appropriate control parameters to achieve a good animation.

6 Conclusion

We have presented a new method for direct control of smoke animation by introducing new terms into the standard flow equations. Each of these terms has a simple closed form, and consequently they allow complex smoke animations to be controlled with little additional expense on top of the cost of an ordinary simulation.

Similarly to previous methods for simulation of smoke [Fedkiw et al. 2001] our simulator is easily extended to support the usual external forces (gravity, buoyancy, etc) in addition to the driving force. It is also possible to add obstacles represented by internal boundary conditions.

In future work it might be interesting to experiment with other terms for driving the smoke through a sequence of target states, as well as other anti-diffusion mechanisms. In particular, we would like to explore a multi-resolution version of our gathering term to make gathering faster, more responsive, and more natural looking.

Another interesting direction would be to explore additional mechanisms for controlling smoke animations. For example, the animator may want to sketch out preferred paths for the smoke to flow along, without the need to generate many intermediate keyframes between the starting and the ending positions.

Acknowledgments

We would like to thank Eri Rubin for providing us with the input animations of the walking mouse and the leaping tiger, and Raz Kupferman for serving as a sounding board throughout this project. Thanks also go to Siggraph's anonymous reviewers for their comments. This work was supported in part by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities and by the Israeli Ministry of Science and Technology.

References

CHORIN, A. J. 1967. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics 2*, 12–16.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (July), 736–744.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Computer Graphics* Proceedings, Annual Conference Series, E. Fiume, Ed., ACM SIGGRAPH, 15–22.

FOSTER, N., AND FEDKIW, R. 2001. Practical animations of liquids. In *Computer Graphics* Proceedings, Annual Conference Series, E. Fiume, Ed., ACM SIGGRAPH, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (Sept.), 471–483.

FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. In *Proceedings CGI '97*, 178–188.

HARLOW, F. H., AND WELCH, J. E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids 8* (Dec.), 2182–2189.

LAMORLETTE, A., AND FOSTER, N. 2002. Structural modeling of flames for a production environment. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (July), 729–735.

LEVEQUE, R. J. 2002. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press.

NGUYEN, D. Q., FEDKIW, R., AND JENSEN, H. W. 2002. Physically based modeling and animation of fire. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (July), 721–728.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press.

STAM, J. 1999. Stable fluids. In *Computer Graphics* Proceedings, Annual Conference Series, A. Rockwood, Ed., ACM SIGGRAPH, 121–128.

TREUILLE, A., MCNAMARA, A., POPOVIC, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (July), 716–723.

TROTTENBERG, U., OOSTERLEE, C., AND SCHÜLLER, A. 2001. *Multigrid*. Academic Press.

WITTING, P. 1999. Computational fluid dynamics in a traditional animation environment. In *Computer Graphics* Proceedings, Annual Conference Series, A. Rockwood, Ed., ACM SIGGRAPH, 129–136.

A A high resolution hyperbolic solver

The solution for hyperbolic conservation laws of the form

$$q_t + (H(q))_x = 0$$

can be approximated numerically by using the *Lax-Wendroff* formula, corrected by a "flux limiter", described in detail in [Leveque 2002]. The idea is improve the accuracy of 1st order upwinding by adding an anti-diffusive correction term. The correction term is modulated by a *flux limiter* in order to prevent oscillations.

More formally, the scheme uses the following update equation

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(\mathscr{H}_{i+\frac{1}{2}}(Q^n) - \mathscr{H}_{i-\frac{1}{2}}(Q^n)).$$

Here Q_i approximates the average density of the conserved quantity q in the cell $[(i-1/2)\Delta x, (i+1/2)\Delta x]$. The *numerical flux function* $\mathscr{H}_{i+\frac{1}{2}}$ approximates the flux $H(q)$ across the boundary between two neighboring cells $(i, i+1)$. The numerical flux function defined by the Lax-Wendroff formula is given by

$$\mathscr{H}^{\mathrm{LW}}(Q^n) = \mathscr{H}^U(Q^n) + \mathscr{C}(Q^n),$$

where $\mathscr{H}^U$ is the *1st order upwinding* flux function defined by

$$\mathscr{H}^U_{1+\frac{1}{2}}(Q^n) = \begin{cases} s_{i+\frac{1}{2}} Q_i^n & s_{i+\frac{1}{2}} > 0 \\ s_{i+\frac{1}{2}} Q_{i+1}^n & s_{i+\frac{1}{2}} < 0 \end{cases}$$

Here $s_{i+\frac{1}{2}}$ approximates the *flux speed* $\partial H/\partial q$ at $\Delta x(i+\frac{1}{2})$, and $\mathscr{C}(Q^n)$ is an anti-diffusive correction term given, in its limited form, by

$$\mathscr{C}(Q^n) = (1 - \frac{\Delta t}{\Delta x}|s_{i+\frac{1}{2}}|)\, mml(Q^n)_{i+\frac{1}{2}}$$

where *mml* is a flux limiter (similar to the commonly used *minmod* limiter [Leveque 2002]) given by

$$mml(Q^n)_{i+\frac{1}{2}} = sgn \cdot min\{|Q_{i+2}^n - Q_{i+1}^n|, |Q_{i+1}^n - Q_i^n|, |Q_i^n - Q_{i-1}^n|\}$$

where

$$sgn = \begin{cases} 1 & Q_{i+2}^n - Q_{i+1}^n, Q_{i+1}^n - Q_i^n, Q_i^n - Q_{i-1}^n > 0 \\ -1 & Q_{i+2}^n - Q_{i+1}^n, Q_{i+1}^n - Q_i^n, Q_i^n - Q_{i-1}^n < 0 \\ 0 & \text{otherwise} \end{cases}$$

As this scheme is based on the Lax-Wendroff formula, it is subject to a time step restriction: $\Delta t < \Delta x/s$, where s is the maximal flux speed.

We use this scheme to approximate the advection terms in the smoke and momentum equations, as well as the smoke gathering term. In the case of advection, the flux speed is simply the velocity along an axis. Therefore, when advecting a quantity along a particular axis, we need the proper velocity component, approximated on the faces perpendicular to that axis. In the case of smoke advection, the cells around the smoke variables coincide with the computational grid cells, so the flux speed is simply given by the velocity variables. When advecting the momentum components U, V, W, we obtain the flux speed on the boundaries between the cells around these variables by the proper averaging.

In the gathering case the flux speed should be evaluated on the faces of computational grid cells:

$$s_{i+\frac{1}{2}} = \mathscr{A}(\tilde{\rho}^*)_{i+\frac{1}{2},j,k} \mathscr{D}_x(\rho - \rho^*)_{i+\frac{1}{2},j,k}.$$