EUROGRAPHICS 2008 / G. Drettakis and R. Scopigno
(Guest Editors)

Volume 27 (2008), Number 2

A Semi-Lagrangian CIP Fluid Solver without Dimensional Splitting

Doyub Kim[†1] Oh-young Song[‡2] and Hyeong-Seok Ko[§1]

[1]Seoul National University, Korea
[2]Sejong University, Korea

Abstract

In this paper, we propose a new constrained interpolation profile (CIP) method that is stable and accurate but requires less amount of computation compared to existing CIP-based solvers. CIP is a high-order fluid advection solver that can reproduce rich details of fluids. It has third-order accuracy but its computation is performed over a compact stencil. These advantageous features of CIP are, however, diluted by the following two shortcomings: (1) CIP contains a defect in the utilization of the grid data, which makes the method suitable only for simulations with a tight CFL restriction; and (2) CIP does not guarantee unconditional stability. There have been several attempts to fix these problems in CIP, but they have been only partially successful. The solutions that fixed both problems ended up introducing other undesirable features, namely increased computation time and/or reduced accuracy. This paper proposes a novel modification of the original CIP method that fixes all of the above problems without increasing the computational load or reducing the accuracy. Both quantitative and visual experiments were performed to test the performance of the new CIP in comparison to existing fluid solvers. The results show that the proposed method brings significant improvements in both accuracy and speed.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

The visual quality of a fluid simulation heavily depends on the accuracy of the advection solver. Since the graphics community became aware of this problem, developing an accurate advection solver has been a primary concern. Attempts have been made to develop high-order advection solvers in the Eulerian framework. These efforts have yielded methods such as the monotonic cubic spline method [FSJ01], the constrained interpolation profile (CIP) method [TFK*03, SSK05], and the back and forth error compensation and correction (BFECC) method [KLLR05, KLLR07]. Hybrid methods, which combine the Lagrangian and Eulerian frameworks, have also been explored. Examples of these techniques are the particle level set method [EMF02], the vortex particle method [SRF05], and derivative particles [SKK07]. What this paper develops is the first kind; It develops a fast, stable, but accurate advection solver in the Eulerian framework. To achieve improved performance, we modified the CIP scheme. Although the proposed method is a purely Eulerian advection solver, it can freely combine with the Lagrangian framework. In fact, it can be used to bring improvements to the above hybrid methods.

Efforts to develop high-order (Eulerian) advection schemes have been led by the computational fluid dynamics (CFD) community. Essentially non-oscillatory (ENO) and weighted ENO (WENO) [OF02] methods are widely used high-order methods in CFD. Another advection scheme based on monotonic cubic-spline polynomials has also been proposed in the graphics field [FSJ01]. In both the ENO/WENO and cubic-spline polynomial approaches, however, the computations are performed over wide stencils, which becomes problematic when the simulation domain

† kim@graphics.snu.ac.kr
‡ oysong@sejong.ac.kr
§ ko@graphics.snu.ac.kr

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

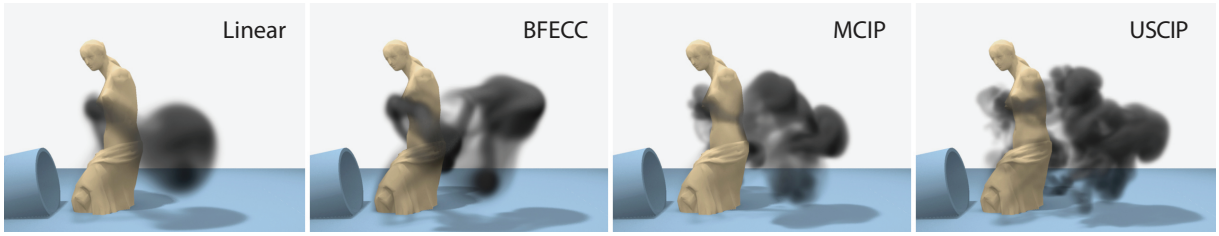

Figure 1: *In this smoke simulation, the linear semi-Lagrangian model took 5.8, BFECC with artificial diffusion [KLL*07] took 29.6, MCIP took 26.92, and our new CIP model (USCIP) took 11.81 seconds per frame to compute the advection. For the whole simulation, the linear model took 41.36, BFECC took 65.48, MCIP took 62.28, and USCIP took 45.39 seconds per frame. The grid resolution was 135×90×90.*

contains complex inner boundaries. Moreover, it is non-trivial to employ adaptive grids when wide stencils are used.

Meanwhile, Yabe and Aoki [YA91, YIW*91] invented a third-order advection scheme, the CIP method, which works with a compact stencil. Unfortunately, this method suffered from instabilities, prompting the development of a modified method known as rational CIP (RCIP) [XYNI96a, XYNI96b]; however, although RCIP is more stable than CIP, it is still not unconditionally stable. Song et al. [SSK05] proposed an unconditionally stable variation of CIP, the monotonic CIP method (MCIP). For the semi-Lagrangian backtracking, they used a dimensional splitting approach based on a tri-cubic interpolation composed of a number of 1D MCIP interpolations. Although MCIP is unconditionally stable, the dimensional splitting significantly increases the computational load, and causes additional numerical dissipation compared to unsplit CIPs.

This paper develops a stable CIP method that does not employ dimensional splitting. The new method, which is based on the semi-Lagrangian method, is unconditionally stable. It runs faster than MCIP and BFECC, but produces results that are clearly less diffusive. (See Figures 1 and 2.)

2. Related Work

In the computer graphics field, fluid animation technique based on full 3D Navier–Stokes equations was first introduced by Foster and Metaxas [FM96]. Subsequently, Stam [Sta99] introduced an unconditionally stable fluid solver based on the semi-Lagrangian advection method.

The first-order semi-Lagrangian advection method uses linear interpolation, which is a source of numerical diffusion. Several high-order Eulerian advection schemes have been proposed to address this problem. Fedkiw et al. [FSJ01] introduced the use of monotonic cubic spline interpolation for the semi-Lagrangian process. Song et al. [SSK05] introduced the monotonic CIP method, and subsequently extended it to the octree data structure [SKK07]. Kim et al. [KLLR05, KLLR07] introduced the BFECC method, which has recently been analyzed by Selle et al. [SFK*07].

To overcome the fundamental drawback associated with grid-based interpolation, several hybrid approaches have proposed that combine the Eulerian and Lagrangian frameworks. Improving the accuracy of surface tracking is another important issue in fluid simulation. To achieve accurate surface tracking in liquid animation, Enright et al. [EMF02] developed the particle level set method, and Mihalef et al. [MMS07] proposed the marker level set method. Velocity also suffers from numerical diffusion, and harms visual realism. For simulating fluids with swirls, Selle et al. [SRF05] embedded vortex particles to the fluid solver. Zhu and Bridson [ZB05] introduced the FLIP method to the graphics community, which performs the advection part in terms of massless particles. Song et al. [SKK07] developed the derivative particle method, which is a combination of the FLIP and CIP methods.

3. Original CIP Method

The method developed in this paper is devised so as to fix the problems associated with the original CIP and MCIP methods. Hence, we describe the original CIP and MCIP in Sections 3 and 4, respectively, before presenting our new CIP method in Section 5

The original version of CIP was introduced in 1991 by Yabe and Aoki [YA91, YIW*91]. The key idea of this method is to advect not only the physical quantities but also their derivatives. In general, the advection equation can be written as

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0, \tag{1}$$

where ϕ is the physical quantity that is being advected. In 1D, differentiating equation (1) with respect to the spatial variable x gives

$$\frac{\partial \phi_x}{\partial t} + u\frac{\partial \phi_x}{\partial x} = -u_x\frac{\partial \phi}{\partial x}, \tag{2}$$

which can be used to predict the evolution of ϕ_x over time. For solving equation (1), the CIP method uses the semi-Lagrangian method: for simplicity, we assume the grid size

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

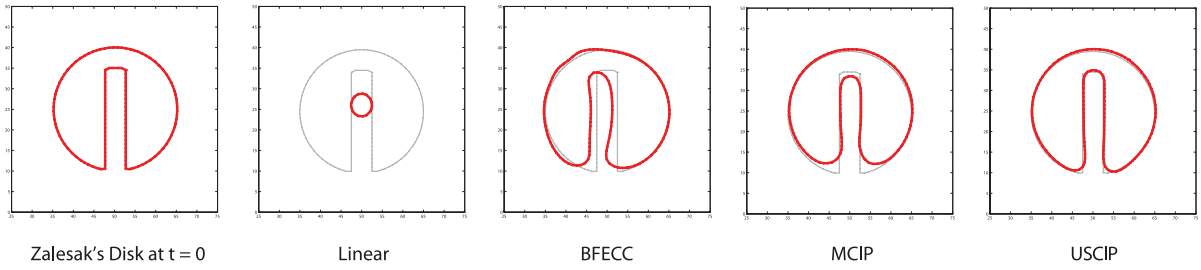

Figure 2: *Zalesak's disk experiment: After one revolution on a 100×100 grid cell computational domain.*

$(x_{i+1} - x_i)$ is 1. If p is the back-tracked position, then its ϕ value is approximated with the cubic-spline interpolation

$$\Phi(p) = [(C_0 p + C_1)p + \phi_{x_i}]p + \phi_i, \tag{3}$$

where the coefficients C_0 and C_1 are given in terms of the ϕ and ϕ_x values of grid points

$$C_0 = \phi_{x_i} + \phi_{x_{i+1}} - 2(\phi_{i+1} - \phi_i) \tag{4}$$

$$C_1 = 3(\phi_{i+1} - \phi_i) - 2\phi_{x_i} - \phi_{x_{i+1}}. \tag{5}$$

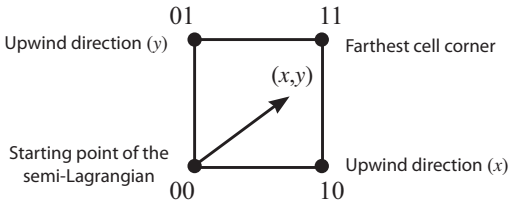

Figure 3: *Indexing of 2D CIP interpolation.*

Extending this method to two and three dimensions, however, turns out not to be straightforward. To extend CIP to higher dimensions, Yabe and Aoki introduced 2D and 3D polynomials [YIW*91]. For 2D, they use the polynomial

$$\Phi(x,y) = \sum_{0 \le i+j \le 3} C_{ij} x^i y^j. \tag{6}$$

The ten coefficients of the above polynomial are determined from four known physical values and six derivative values at the cell corners. It is critical to note where they took the derivative values. They took x and y directional derivatives from only three cell corners, specifically the two upwind directions and the starting point of the semi-Lagrangian, as shown in Figure 3. Since the construction of the above polynomial does not utilize the derivative information at the farthest cell corner, the method is accurate only when the back-tracked point falls near the starting point of the semi-Lagrangian advection. This can be problematic for simulations with large CFL numbers. Another critical problem of the original CIP methods is that they can generate instabilities. Even in the case of 1D CIP interpolation, stability is not guaranteed. The Hermite interpolating polynomial, which is defined with the values and their derivatives at the end points, can easily generate overshooting profiles.

4. Monotonic CIP Method

Song et al. [SSK05] proposed a variation of the original CIP method, namely the monotonic CIP (MCIP). To ensure stability, MCIP uses a modified version of the grid point derivatives if the profiles of equation (3) can potentially have overshoots. Song et al. derived the sufficient condition for the grid point derivatives that guarantees a monotonic profile. The MCIP method is unconditionally stable. However, the derivative clamping in MCIP can over-stabilize the situation; in cases where the original CIP does not generate instabilities, MCIP tends to give slightly more diffusive simulation results compared to the original CIP method.

Song et al. [SSK05] extended the 1D MCIP method to 2D and 3D. They obtained the higher dimensional MCIPs by cascading the 1D MCIP solver. This dimensional splitting is described in detail in [SSK05]. Since the 2D/3D MCIPs are composed of monotonic interpolations, they are also unconditionally stable. With the dissipation-suppressing nature inherited from the original CIP and the unconditional stability achieved by Song et al., MCIP could perform 2D simulations of water in real-time.

Although the dimensional splitting brought stability to the MCIP method, it has two major drawbacks. Firstly, the dimensional splitting of MCIP leads to a higher computational load compared to unsplit CIP. In 2D, six cubic-spline interpolations must be performed for a single semi-Lagrangian access [SSK05]: two along the x-axis for ϕ and ϕ_x, two along the x-axis for ϕ_y and ϕ_{xy}, one along the y-axis for ϕ and ϕ_y, and one along the y-axis for ϕ_x and ϕ_{xy}. In 3D, 27 cubic-spline interpolations are required for a single access (since the second and third derivatives must be interpolated as well). Also, MCIP uses a large number of condition statements (if-else) to keep a monotonic profile. Song et al. [SKK07] have shown that, in the 3D octree data-structure,

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

the computation time for MCIP is 60% higher than that of linear advection[†].

The second drawback associated with the dimensional splitting of MCIP is numerical error. The results produced using the split-interpolation are not identical to those produced using the unsplit-interpolation; the results of the split-interpolation depend on the axis direction. Moreover, Xiao et al. [XYNI96a] have shown that dimensional-splitting exhibits numerical diffusion when simulating shear motion. Also, we note that the split-CIP-interpolation requires second and third derivatives that must be calculated by central differencing. This represents another source of numerical diffusion, and calls for a non-negligible amount of extra computation. More importantly, the use of central differencing harms the compactness of the CIP method.

From the above, it is apparent that unsplit-CIP-interpolation is more attractive than CIP interpolation with dimensional splitting. However, although unsplit-CIP-interpolation has existed since the birth of the CIP method, a stable unsplit-CIP-interpolation that can be used for any (unrestricted) semi-Lagrangian advection has yet to be developed.

5. Unsplit Semi-Lagrangian CIP Method

This paper develops an unsplit semi-Lagrangian CIP (USCIP) method, where the words 'semi-Lagrangian' in the name bears the stability. Our proposed technique should be applicable to simulations without any CFL restrictions. To develop USCIP, we go back to Yabe and Aoki's original 2D and 3D polynomials and make necessary modifications. A fundamental deviation we make from the original CIP is that we utilize all the derivative information for each cell. In 2D, a cell has 12 known values: ϕ, ϕ_x, and ϕ_y at the four corners. Another deviation from the original CIP is that we include two additional terms, x^3y and xy^3, in the polynomial; specifically, the 2D polynomial we use for USCIP is

$$\Phi(x,y) = \sum_{0\le i+j\le 3} C_{ij}x^i y^j + C_{31}x^3y + C_{13}xy^3. \qquad (7)$$

We included the two extra terms because of the mismatch between the number of known values (12) and the number of terms (10) in the third-order 2D polynomial. There are two options for overcoming this mismatch arising from the use of all the known values: to formulate the coefficient-finding task as an over-constrained problem and find the least-squares solution; or to insert extra terms to match the number of known values. We chose the latter option in this paper. We did not follow the first option because when interpolation is performed with the least-squares solution, (1) the interpolated result at the corner will not be identical to the known value at that corner, and (2) it will not be C^0 continuous across the cell boundaries.

The forms of the two added terms were decided according to the following three principles: 1) the new terms should not create any asymmetry, i.e., if $x^m y^n$ is added, then $x^n y^m$ must also be added; 2) the new terms should contain both x and y, since such terms can reflect off-axis motion such as rotation and shear better than decoupled terms; and 3) among the terms that satisfy the first and second principles, the lowest order terms should be chosen to prevent any unnecessary wiggles. The terms that pass all three criteria are x^3y and xy^3.

The coefficients of the polynomial can be computed in a manner similar to that described by Yabe and Aoki [YIW*91]. Let ϕ_{00}, ϕ_{10}, ϕ_{01}, and ϕ_{11} be the physical quantities at each cell corner. Let $\phi_x = \partial\phi/\partial x$, and $\phi_y = \partial\phi/\partial y$. Let ϕ_{x00}, ϕ_{x10}, ϕ_{x01}, ϕ_{x11}, ϕ_{y00}, ϕ_{y10}, ϕ_{y01}, and ϕ_{y11} be the derivative values at each cell corner. Then, the coefficients $C_{00}\ldots C_{31}$ are uniquely given by,

$$\begin{aligned}
C_{00} &= \phi_{00}\\
C_{10} &= \phi_{x00}\\
C_{01} &= \phi_{y00}\\
C_{20} &= 3(\phi_{10}-\phi_{00}) - \phi_{x10} - 2\phi_{x00}\\
C_{02} &= 3(\phi_{01}-\phi_{00}) - \phi_{y01} - 2\phi_{y00}\\
C_{30} &= -2(\phi_{10}-\phi_{00}) + \phi_{x10} + \phi_{x00}\\
C_{03} &= -2(\phi_{01}-\phi_{00}) + \phi_{y01} + \phi_{y00}\\
C_{21} &= 3\phi_{11} - 2\phi_{x01} - \phi_{x11}\\
&\quad - 3(C_{00}+C_{01}+C_{02}+C_{03}) - C_{20}\\
C_{31} &= -2\phi_{11} + \phi_{x01} + \phi_{x11}\\
&\quad + 2(C_{00}+C_{01}+C_{02}+C_{03}) - C_{30}\\
C_{12} &= 3\phi_{11} - 2\phi_{y10} - \phi_{y11}\\
&\quad - 3(C_{30}+C_{20}+C_{10}+C_{00}) - C_{02}\\
C_{13} &= -2\phi_{11} + \phi_{y10} + \phi_{y11}\\
&\quad + 2(C_{00}+C_{10}+C_{20}+C_{30}) - C_{03}\\
C_{11} &= \phi_{x01} - C_{13} - C_{12} - C_{10}. \qquad (8)
\end{aligned}$$

For the 3D case, the coefficients are presented in Appendix A.

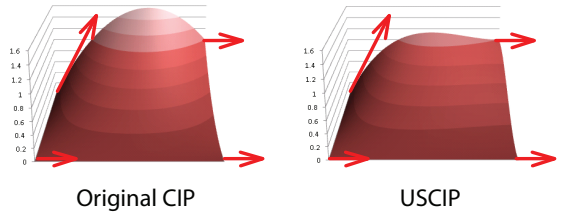

Figure 4: *Interpolated profile of the original CIP and USCIP in 2D. Red arrows correspond to ϕ_x.*

Figure 4 visualizes the effect of utilizing the derivative in-

[†] Since most discussions in this paper are made in the context of semi-Lagrangian advection, we will refer to first-order semi-Lagrangian advection simply as linear advection.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

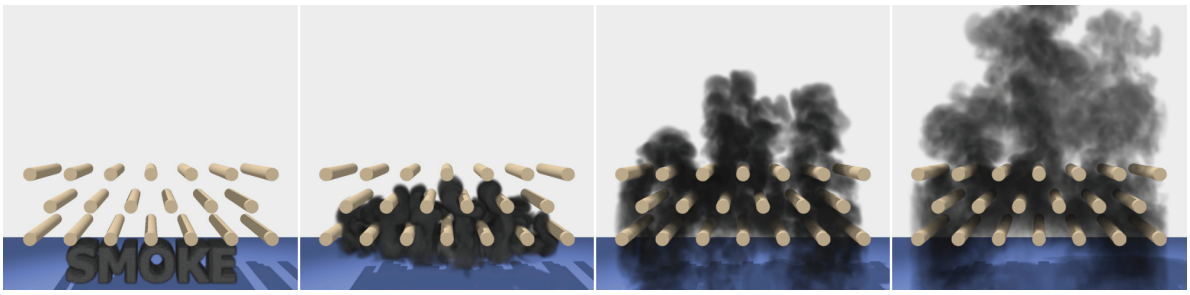

Figure 5: *Simulation of rising smoke passing through obstacles. The grid resolution was 160×220×80.*

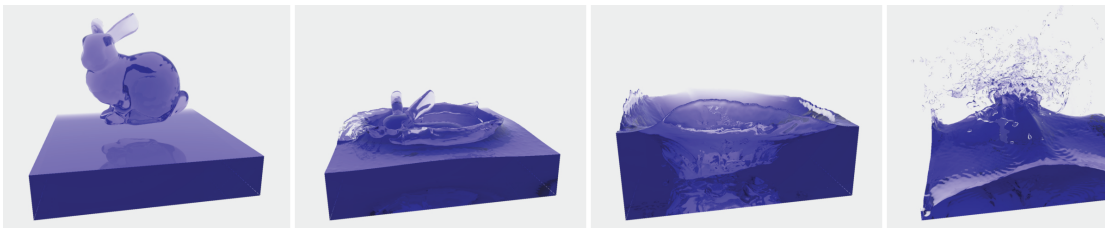

Figure 6: *Simulation of a bunny-shaped water chunk dropping onto still water. The grid resolution was* 150^3.

formation at the farthest cell corner (the back right corner in the figure). At every corner, $\phi_y = 0$. At the back right corner, $\phi = 1$, $\phi_x = 0$, and at the back left corner, $\phi = 0$, $\phi_x = 5$, while $\phi = \phi_x = 0$ at the remaining corners. We can see that the original CIP forms a steep slope around the farthest corner, whereas USCIP reflects all the derivative information in the profile.

Although utilization of the derivative information at the farthest cell corner helps improve the stability, it does not guarantee that the interpolated value will always be bounded by the grid point values. Thus, we need to make a provision to keep the USCIP stable. One option would be to follow the approach taken in MCIP; that is, we could find the sufficient condition for the derivatives which makes the profile monotonic for an arbitrary direction. However, we do not take this approach for the following reasons: (1) finding the sufficient condition becomes a 24-degree-of-freedom optimization problem in 3D since we have three derivatives for each of the eight cell corners, and (2) clamping the derivatives into the sufficient condition might lead to over-stabilization in some cases that are not overshoots.

In USCIP, we perform a very simple clamping: when the interpolated result is larger/smaller than the maximum/minimum of the cell node values, we replace the result with the maximum/minimum value. This *delayed clamping* procedure, which is similar to the one used in the unconditionally stable MacCormack scheme [SFK*07], guarantees unconditional stability without introducing unnecessary over-stabilization.

USCIP works on compact stencils since it does not need to calculate high-order derivatives. This is an important improvement over MCIP. Obviating the calculation of high-order derivatives also reduces the computation time.

Although constructing and interpolating with a high-order unsplit polynomial is more complicated than working with 1D split polynomials, since the split-CIP involves multiple interpolations, overall USCIP requires fewer operations than MCIP. According to our implementation, MCIP performs 693 (21×27+126) operations for a 3D interpolation; that is, 27 1D interpolations must be performed for a 3D tri-cubic interpolation, and each 1D interpolation involves 21 operations. Additionally, MCIP must compute second and third derivatives, which involves 126 operations. By contrast, USCIP performs 296 operations for a 3D interpolation, which corresponds to only 43% of the total operation count needed for MCIP.

6. Experimental Results

The simulations reported in this section were performed on a PowerMac G5 2.5 GHz with 5.5 GB of memory. For the simulations on Zalesak's disk and smoke, the linear, BFECC, and MCIP advections were also performed for comparison. BFECC and MCIP were implemented according to [KLL*07] and [SSK05], respectively. In every simulation, a uniform grid was used, and the CFL number was restricted to two. No vortex reinforcement method such as that described in [FSJ01] or [SRF05] was used in any of the experiments. Rendering was performed using our in-house ray-tracer. The pseudo-code for our 2D fluid solver is described in Table 2. Although the pseudo-code does not include the density or

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

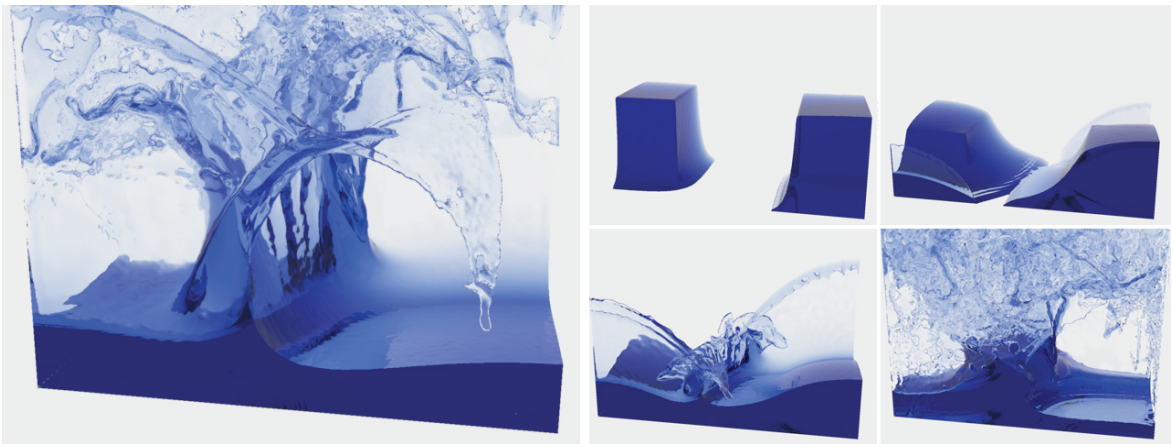

Figure 7: *Simulation of a dam breaking with 200×150×100 grid resolution.*

level set implementation, it is straightforward to extend it to smoke or liquid simulator.

6.1. Rigid Body Rotation of Zalesak's Disk

We performed Zalesak's disk experiment [Zal79] on a 100×100 resolution grid. The contour of the disk was tracked via the level set field. This experiment is not designed to measure the interface tracking capability. (Interface tracking can be easily improved by applying hybrid techniques such as those in [ELF03], [SKK07] or [MMS07].) Rather, the experiment focuses on measuring the anti-dissipation capabilities of purely Eulerian advection methods. For the same reason, we did not perform reinitialization of the level set, as in [SFK*07].

We rotated the disk using four advection schemes: linear (first-order semi-Lagrangian), BFECC, MCIP, and USCIP. The results, shown in Figure 2, indicate that USCIP produces the most accurate result and that, as explained in Sections 4 and 5, USCIP produces a less diffusive result than MCIP.

6.2. Smoke Injected Toward a Statue

In this experiment, smoke is injected toward a statue and vortices are generated behind the statue. As in the previous experiment, we ran the simulation with four advection schemes and measured the computation times. The grid resolution was 135×90×90. Figure 1 shows snapshots taken at the same simulation time in the computations using the four advection schemes. Table 1 summarizes the average computation time for simulating a single frame (not for simulating a single time step). The figure and table show that, among the three high-order schemes, USCIP runs more than twice as fast as BFECC and MCIP but produces a result that is clearly less diffusive.

We note that USCIP ran faster than BFECC. In fact, USCIP involves more operations than BFECC in a single loop, but BFECC performs more loops than USCIP. Thus USCIP takes advantage of the cache hit. Also, BFECC has to perform an artificially designed diffusion process for suppressing noise [KLL*07], which adds more operations. Since USCIP has to store spatial derivatives, it uses three times more memory for each advection. This is not a problem, however, given that it is now possible to mount gigabytes of memories in PCs. Even with uniform grids, USCIP could simulate a fairly complex fluid scene on a single PC.

6.3. Rising Smoke Passing Through Obstacles

Initially the smoke forms the letters of the word 'SMOKE' in 3D. As the smoke rises due to the buoyancy force, it hits obstacles, leading to the generation of many complex swirls inside the domain. The smoke was simulated with USCIP on an 160×220×80 grid. A series of snapshots is shown in Figure 5. This experiment demonstrates that USCIP can generate realistic swirling of smoke under complicated internal boundary conditions without the assistance of vortex reinforcement methods.

6.4. Dropping a Bunny-shaped Water onto Still Water

This experiment and the next one simulate the motion of water, and show that USCIP can be used in hybrid methods. In these two experiments, we employed the particle level set method [EMF02] for tracking the water surfaces. In the current experiment, a bunny-shaped water chunk was dropped onto still water. The grid resolution was 150^3. Figure 6 shows a series of snapshots as the bunny strikes the water surface. USCIP successfully generated complicated small-scale features such as droplets, thin water sheets, and small waves.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

	Linear	BFECC	MCIP	USCIP
Computation Time for Advection Only (sec/frame)	5.80	29.60	26.92	11.81
Computation Time for Total Simulation (sec/frame)	41.36	65.48	62.28	45.39

Table 1: *Computation time for the smoke simulation shown in Figure 1*

6.5. Colliding Water after Dam Breaking

We next used USCIP to simulate a mid-scale dam breaking. The grid resolution was 200×150×100. Two columns of water, each with dimensions 0.6m×0.8m×0.6m, were released to make a violent collision. In this situation, advection is the dominant part in the fluid simulation. As shown in Figure 7, after the collision of two columns of water, thin and tall water sheets were developing and then lost their momentum near the ceiling. After that, the sheets were falling in shapes of many tiny droplets due to the gravity. This experiment indicates that USCIP can reproduce the detailed movements of fast fluids.

6.6. Vorticity Preservation Test

Although our method is purely Eulerian, we compared USCIP with FLIP [ZB05], which is a particle-grid hybrid method. We initially set the velocities of a 1m×1m domain to the single vortex field which is defined by the stream function

$$\psi = \frac{1}{\pi} sin^2(\pi x) sin^2(\pi y). \tag{9}$$

For FLIP, we seeded nine particles for each cell of 100×100 grids. We used Zhu and Bridson's 2D FLIP solver ‡ for the FLIP simulation. As shown in Figure 8, the result generated with FLIP shows noisy curl field. Such a problem is not visible in the result generated with USCIP.

7. Conclusion

In this paper, we presented a new semi-Lagrangian CIP method which is stable, fast, and produces accurate results. By noting the problems associated with the original CIP and MCIP methods, we concluded that a new polynomial containing two additional fourth-order terms could be used for the third-order interpolation. By judiciously choosing the newly introduced terms, the proposed advection technique could reflect all the derivative information stored at the grid points without producing any noticeable artifacts. The proposed technique ran more than twice as fast as BFECC or MCIP but produced results that were clearly less diffusive. Since USCIP works at a fundamental level, it can be applied to existing advanced fluid simulators to enhance their speed, stability, and accuracy.

‡ http://www.cs.ubc.ca/~rbridson

advance_single_time_step () {
  // Non-advection part for $\mathbf{u}(u,v)$ (see [Sta99] for the details).
  $\mathbf{u}^*$ ⇐ add_force / diffuse / project ($\mathbf{u}^n$)

  // Compute RHS of the equation (2) for $\mathbf{u}(u,v)$.
  ∇u^* ⇐ update_derivatives (u^n, ∇u^n, u^*, $\mathbf{u}^n$)
  ∇v^* ⇐ update_derivatives (v^n, ∇v^n, v^*, $\mathbf{u}^n$)

  // Advection part for $\mathbf{u}(u,v)$.
  // Compute equation (1) and LHS of the equation (2).
  $\mathbf{u}^{n+1}$, $\nabla\mathbf{u}^{n+1}$ ⇐ advect_velocity_USCIP ($\mathbf{u}^*$, $\nabla\mathbf{u}^*$)
}

// Update derivatives for f (see [YIW*91] for more details).
update_derivatives (f^n, ∇f^n, f^*, $\mathbf{u}^n$) {
  for each cell {
    $f^*_{x_{i,j}} = f^n_{x_{i,j}} + \frac{1}{2\Delta x}(f^*_{i+1,j} - f^*_{i-1,j} - f^n_{i+1,j} + f^n_{i-1,j})$
      $-\frac{\Delta t}{\Delta x}[f^n_{x_{i,j}}(u^n_{i+1/2,j} - u^n_{i-1/2,j}) + f^n_{y_{i,j}}(v^n_{i+1/2,j} - v^n_{i-1/2,j})]$

    $f^*_{y_{i,j}} = f^n_{y_{i,j}} + \frac{1}{2\Delta y}(f^*_{i,j+1} - f^*_{i,j-1} - f^n_{i,j+1} + f^n_{i,j-1})$
      $-\frac{\Delta t}{\Delta y}[f^n_{x_{i,j}}(u^n_{i,j+1/2} - u^n_{i,j-1/2}) + f^n_{y_{i,j}}(v^n_{i,j+1/2} - v^n_{i,j-1/2})]$
  }
  return ∇f^*
}

Table 2: *Pseudo-code for the 2D USCIP fluid solver.*

8. Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korean government (MOST) (National Research Laboratory M10600000232-06J0000-23210), Ministry of Information and Communication, the Brain Korea 21 Project, the faculty research fund of Sejong University in 2006, Seoul Research and Business Development Program (10557), and Automation and System Research Institute at Seoul National University.

References

[ELF03] ENRIGHT D., LOSSASO F., FEDKIW R.: A fast and accurate semi-lagrangian particle level set method. *Computers and Structures 83* (2003), 479–490.

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.:

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

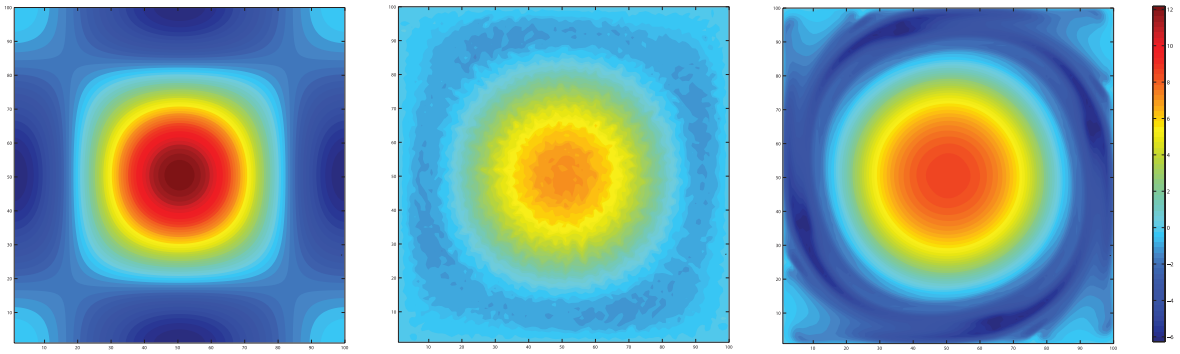

Figure 8: *2D Vorticity preservation test on a 100×100 grids. The left image shows the magnitude of initial vorticity. The middle and the right images show the results of the FLIP and USCIP methods, respectively, at $t = 3.0$ sec. In the images, color represents the magnitude of the vorticity.*

Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3 (2002), 736–744.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical models and image processing: GMIP 58*, 5 (1996), 471–483.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. *Computer Graphics (Proc. ACM SIGGRAPH 2001) 35* (2001), 15–22.

[KLL*07] KIM B., LIU Y., LLAMAS I., JIAO X., ROSSIGNAC J.: Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph. 26*, 3 (2007), 98.

[KLLR05] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Flowfixer: Using bfecc for fluid simulation. In *Eurographics Workshop on Natural Phenomena 2005* (2005).

[KLLR07] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Advections with significantly reduced dissipation and diffusion. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (2007).

[MMS07] MIHALEF V., METAXAS D., SUSSMAN M.: Textured liquids based on the marker level set. In *Eurographics 2007 proceedings* (2007), pp. 457–466.

[OF02] OSHER S., FEDKIW R.: *The Level Set Method and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2002.

[SFK*07] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An unconditionally stable maccormack method. *J. Sci. Comput. in review* (2007).

[SKK07] SONG O.-Y., KIM D., KO H.-S.: Derivative particles for simulating detailed movements of fluids. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 711–719.

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. 24*, 3 (2005).

[SSK05] SONG O.-Y., SHIN H., KO H.-S.: Stable but non-dissipative water. *ACM Trans. Graph. 24*, 1 (2005), 81–97.

[Sta99] STAM J.: Stable fluids. *Computer Graphics (Proc. ACM SIGGRAPH '99) 33*, Annual Conference Series (1999), 121–128.

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. In *Eurographics 2003 proceedings* (2003), pp. 391–400.

[XYNI96a] XIAO F., YABE T., NIZAM G., ITO T.: Constructing a multi-dimensional oscillation preventing scheme for the advection equation by a rational function. *Computer Physics Communications 94*, 2-3 (1996), 103–118.

[XYNI96b] XIAO F., YABE T., NIZAM G., ITO T.: Constructing oscillation preventing scheme for the advection equation by a rational function. *Computer Physics Communications 93*, 1 (1996), 1–12.

[YA91] YABE T., AOKI T.: A universal solver for hyperbolic equations by cubic-polynomial interpolation i. one-dimensional solver. *Computer Physics Communications 66*, 2-3 (1991), 219–232.

[YIW*91] YABE T., ISHIKAWA T., WANG P. Y., AOKI T., KADOTA Y., IKEDA F.: A universal solver for hyperbolic equations by cubic-polynomial interpolation ii. two- and three-dimensional solvers. *Computer Physics Communications 66*, 2-3 (1991), 233–242.

[Zal79] ZALESAK S. T.: Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comp. Phys. 31*, 3 (1979), 335–362.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (2005), 965–972.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

Appendix A: The Coefficients of 3D USCIP

The polynomial used for the 3D USCIP advection is

$$\begin{aligned}\Phi(x,y,z) &= \sum_{0\le i+j+k\le 3} C_{ijk}x^i y^j z^k \\ &+ C_{310}x^3y + C_{301}xz^3 + C_{130}xy^3 \\ &+ C_{031}y^3z + C_{103}z^3x + C_{013}z^3y \\ &+ C_{211}x^2yx + C_{121}xy^2z + C_{112}xyz^2 \\ &+ C_{311}x^3yx + C_{131}xy^3z + C_{113}xyz^3.\end{aligned}$$

Known values are ϕ_{ijk}, ϕ_{xijk}, ϕ_{yijk}, ϕ_{zijk}, where i, j, k are 0 or 1. Let $\Delta x_{jk} = \phi_{x1jk} - \phi_{x0jk}$, $\Delta y_{ik} = \phi_{yi1k} - \phi_{yi0k}$, and $\Delta z_{ij} = \phi_{zij1} - \phi_{zij0}$. Then, the coefficients are given by

$$\begin{aligned}
C_{000} &= \phi_{000},\ C_{100} = \phi_{x000},\ C_{010} = \phi_{y000},\ C_{001} = \phi_{z000} \\
C_{110} &= \phi_{x010} - \phi_{x000} - (\Delta y_{10} - \Delta y_{00}) + (\phi_{y100} - \phi_{y000}) \\
C_{011} &= \phi_{y001} - \phi_{y000} - (\Delta z_{01} - \Delta z_{00}) + (\phi_{z010} - \phi_{z000}) \\
C_{101} &= \phi_{z100} - \phi_{z000} - (\Delta x_{01} - \Delta x_{00}) + (\phi_{x001} - \phi_{x000}) \\
C_{200} &= 3\Delta x_{00} - \phi_{x100} - 2\phi_{x000} \\
C_{020} &= 3\Delta y_{00} - \phi_{y010} - 2\phi_{y000} \\
C_{002} &= 3\Delta z_{00} - \phi_{z001} - 2\phi_{z000} \\
C_{111} &= (\phi_{x011} - \phi_{x010} - \phi_{x001} + \phi_{x000}) \\
&+ (\phi_{y101} - \phi_{y100} - \phi_{y001} + \phi_{y000}) \\
&+ (\phi_{z110} - \phi_{z100} - \phi_{z010} + \phi_{z000}) - 2(\phi_{111} - A) \\
C_{210} &= 3(\Delta x_{10} - \Delta x_{00}) - 2(\phi_{x010} - \phi_{x000} - \phi_{x110} + \phi_{x100}) \\
C_{201} &= 3(\Delta x_{01} - \Delta x_{00}) - 2(\phi_{x001} - \phi_{x000} - \phi_{x101} + \phi_{x100}) \\
C_{120} &= 3(\Delta y_{10} - \Delta y_{00}) - 2(\phi_{y100} - \phi_{y000} - \phi_{y110} + \phi_{y010}) \\
C_{021} &= 3(\Delta y_{01} - \Delta y_{00}) - 2(\phi_{y001} - \phi_{y000} - \phi_{y011} + \phi_{y010}) \\
C_{102} &= 3(\Delta z_{10} - \Delta z_{00}) - 2(\phi_{z100} - \phi_{z000} - \phi_{z101} + \phi_{z001}) \\
C_{012} &= 3(\Delta z_{01} - \Delta z_{00}) - 2(\phi_{z010} - \phi_{z000} - \phi_{z011} + \phi_{z001}) \\
C_{300} &= \phi_{x100} + \phi_{x000} - 2\Delta x_{00} \\
C_{030} &= \phi_{y010} + \phi_{y000} - 2\Delta y_{00} \\
C_{003} &= \phi_{z001} + \phi_{z000} - 2\Delta z_{00} \\
C_{310} &= \phi_{x110} - \phi_{x100} + \phi_{x010} - \phi_{x000} - 2(\Delta x_{10} - \Delta x_{00}) \\
C_{301} &= \phi_{x101} - \phi_{x100} + \phi_{x001} - \phi_{x000} - 2(\Delta x_{01} - \Delta x_{00}) \\
C_{130} &= \phi_{y110} - \phi_{y010} + \phi_{y100} - \phi_{y000} - 2(\Delta y_{10} - \Delta y_{00}) \\
C_{031} &= \phi_{y011} - \phi_{y010} + \phi_{y001} - \phi_{y000} - 2(\Delta y_{01} - \Delta y_{00}) \\
C_{103} &= \phi_{z101} - \phi_{z000} + \phi_{z100} - \phi_{z000} - 2(\Delta z_{10} - \Delta z_{00}) \\
C_{013} &= \phi_{z011} - \phi_{z000} + \phi_{z010} - \phi_{z000} - 2(\Delta z_{01} - \Delta z_{00}) \\
C_{211} &= 3(\phi_{111} - A) - (\phi_{x111} - \phi_{x110} - \phi_{x101} + \phi_{x100}) \\
&- 2(\phi_{x011} - \phi_{x010} - \phi_{x001} + \phi_{x000}) \\
C_{121} &= 3(\phi_{111} - A) - (\phi_{y111} - \phi_{y110} - \phi_{y011} + \phi_{y010}) \\
&- 2(\phi_{y101} - \phi_{y100} - \phi_{y001} + \phi_{y000}) \\
C_{112} &= 3(\phi_{111} - A) - (\phi_{z111} - \phi_{z101} - \phi_{z011} + \phi_{z001}) \\
&- 2(\phi_{z110} - \phi_{z100} - \phi_{z010} + \phi_{z000}) \\
C_{311} &= (\phi_{x111} - \phi_{x110} - \phi_{x101} + \phi_{x100}) \\
&+ (\phi_{x011} - \phi_{x010} - \phi_{x001} + \phi_{x000}) - 2(\phi_{111} - A) \\
C_{131} &= (\phi_{y111} - \phi_{y110} - \phi_{y011} + \phi_{y010}) \\
&+ (\phi_{y101} - \phi_{y100} - \phi_{y001} + \phi_{y000}) - 2(\phi_{111} - A) \\
C_{113} &= (\phi_{z111} - \phi_{z101} - \phi_{z011} + \phi_{z001}) \\
&+ (\phi_{z110} - \phi_{z100} - \phi_{z010} + \phi_{z000}) - 2(\phi_{111} - A) \\
A &= \phi_{100} + \phi_{y100} + \phi_{z100} + C_{011} + C_{020} + C_{002} + C_{120} + C_{021} \\
&+ C_{102} + C_{012} + C_{030} + C_{003} + C_{130} + C_{031} + C_{103} + C_{013}.
\end{aligned}$$

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.