

Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques

Geoffrey Irving*
Stanford University
Pixar Animation Studios

Eran Guendelman*
Stanford University

Frank Losasso*
Stanford University
Industrial Light + Magic

Ronald Fedkiw*
Stanford University
Industrial Light + Magic

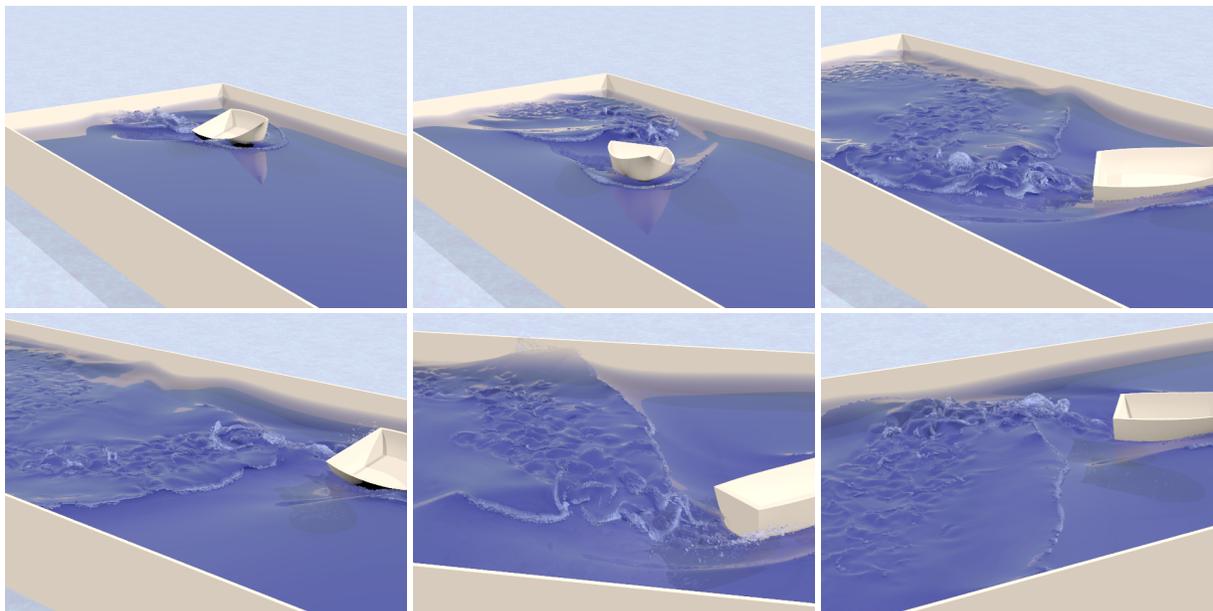


Figure 1: Simulation of a wake behind a kinematically scripted boat (1500×300 horizontal resolution).

Abstract

We present a new method for the efficient simulation of large bodies of water, especially effective when three-dimensional surface effects are important. Similar to a traditional two-dimensional height field approach, most of the water volume is represented by tall cells which are assumed to have linear pressure profiles. In order to avoid the limitations typically associated with a height field approach, we simulate the entire top surface of the water volume with a state of the art, fully three-dimensional Navier-Stokes free surface solver. Our philosophy is to use the best available method near the interface (in the three-dimensional region) and to coarsen the mesh away from the interface for efficiency. We coarsen with tall, thin cells (as opposed to octrees or AMR), because they maintain good resolution horizontally allowing for accurate representation of bottom topography.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling;

Keywords: water, adaptive simulation, rivers, streams, Navier-Stokes equations

*e-mail: {irving,erang,losasso,fedkiw}@cs.stanford.edu

1 Introduction

Scenes involving stormy seas, sudden floods or cascading rapids provide some of the most spectacular visual effects shots in feature films (e.g. “The Day After Tomorrow” [Iversen and Sakaguchi 2004]), and are invariably expensive whether they are simulated via computer or hundreds of thousands of gallons of real water. Since water is opaque at large scales, its appearance is governed by a surface layer while the interior flow is “visible” only through its effect on the surface.

Fully three-dimensional Navier-Stokes solvers produce stunning results, but do not scale well to large bodies of water when the simulation is carried out on a uniform Cartesian grid. Although the situation can be improved significantly by coarsening away from the surface with an octree as in [Losasso et al. 2004], there are two major drawbacks to this approach. First, the current octree approaches do not make full use of the highly accurate MAC grid method as in [Enright et al. 2002], instead relying too much on nodal values for interpolation resulting in increased numerical dissipation. Of course, this can be fixed by coarsening the octree away from the interface, and otherwise applying a standard MAC solver in the uniform cells near the water surface. Second, and more importantly (especially since we propose no remedy), large octree cells cannot represent bottom topography. Even if one refined near the bottom of the domain, the large octree cells in the middle of the water filter out horizontal detail making the surface simulation unaware of any rich structure below. In contrast, methods based on height fields (such as the shallow water equations [Kass and Miller 1990; O’Brien and Hodgins 1995]) use tall cells with detailed refinement in the horizontal directions, thus capturing the effects of complex bottom topography rather well. Unfortunately, these methods do not support overturning or other interesting three-dimensional behavior.

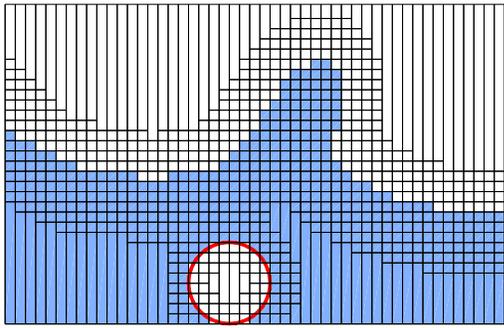


Figure 2: We use uniform cells near objects and within a specified optical depth of the surface, and coarsen with tall cells elsewhere.

Thus, we place the following requirements on our method. First, we need detailed three-dimensional behavior near the interface, down to the depth at which turbulent motion directly affects the look of the surface. This “optical depth” for simulation will usually be greater than the visible optical depth, since interesting turbulence will cause the visible water to continuously mix with the water in a larger layer underneath. Second, the large unseen region of the liquid should be represented as efficiently as possible without losing plausible bulk motion and important details such as bottom topography. The first requirement is satisfied by using a state of the art, uniform MAC grid Navier-Stokes solver near the interface (as in [Enright et al. 2002]). This also allows for the addition of any other technique that works on a uniform grid, such as vortex particles [Selle et al. 2005]. Outside of this surface layer, we maintain the same resolution in the horizontal directions but coarsen in the vertical direction obtaining tall thin cells that reach down to the bottom of the domain (satisfying the second requirement). Since the effects of bottom topography diminish with increasing depth, the method is most useful in the shallow water regime.

One advantage of this hybrid approach is that it reduces to exactly the standard MAC discretization under full refinement. In contrast, the octree method of [Losasso et al. 2004] suffers from increased dissipation due to repeated back and forth averaging even when fully refined. While this dissipation was partially reduced in [Guedelman et al. 2005] using FLIP rather than PIC averaging (see also [Zhu and Bridson 2005]), our experience with uniform grids indicates that the full MAC method still produces higher quality results.

Although a grid of this type could be represented with pointer structures (as in [Whitaker 1998]) connecting all the tall and short cells together in the required fashion, this could prove rather inefficient. Luckily, we can leverage recent work on Run-Length-Encoded (RLE) grids by [Houston et al. 2004; Breen et al. 2004; Wiebe and Houston 2004; Nielsen and Museth 2005; Houston et al. 2005; Houston et al. 2006] (note that [Curless and Levoy 1996] applied RLE to volumes and [Bridson 2003] suggested RLE level sets). Their fluid simulator solves the level set on a narrow band near the interface, but uses a uniform MAC discretization inside the liquid for the fluid solver with long cells allowed only in the air. This is analogous to previous local level set and narrow band methods such as [Peng et al. 1999] and [Adalsteinsson and Sethian 1995]. Thus, their method improves the cost of the level set algorithm and removes the need to store and check grid cells deep inside the air, but does not reduce the time spent solving for advection and pressure in the fluid itself, which is typically most of the computational cost. In contrast, we use tall cells in the water, which greatly reduces the computational cost outside the band of cells near the interface. This requires novel methods both for discretizing tall cells full of liquid as well as for coupling these cells to the standard uniform cells. Our tall cells only occur far from the interface, and thus

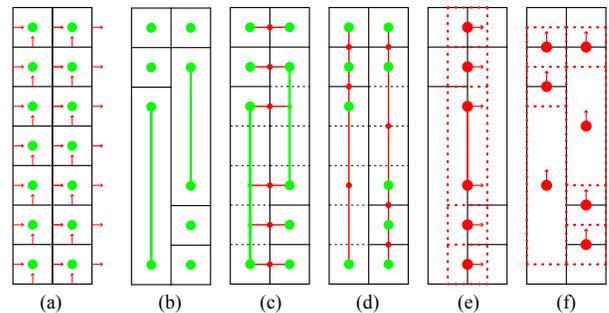


Figure 3: (a) pressure and velocity for a uniform MAC grid (b) pressure in uniform and tall cells (c) horizontal pressure derivatives (d) vertical pressure derivatives (e) horizontal velocities co-located with horizontal pressure derivatives (f) vertical velocities co-located with vertical pressure derivatives.

do not participate in the particle level set method. However, we do require methods for advecting the velocity field and solving for the pressure in order to make the velocity divergence free. Note that all our tall cells are vertical as is typical for shallow water methods, because it is reasonable to assume linear pressure variation in the vertical direction but not in the horizontal directions.

2 Previous Work

The three-dimensional Navier-Stokes equations were popularized with the work of [Foster and Metaxas 1997b; Stam 1999; Fedkiw et al. 2001]. These equations were combined with methods for simulating a water surface in [Foster and Metaxas 1996; Foster and Metaxas 1997a; Foster and Fedkiw 2001], and we use the particle level set approach of [Enright et al. 2002] as the method of choice in the three-dimensional surface layer of water. [Baxter et al. 2004b] used a conservative heightfield fluid model in an interactive painting system. Other work on water and liquids includes viscoelastic fluids [Goktekin et al. 2004], solid fluid coupling [Carlson et al. 2004; Guendelman et al. 2005], control [McNamara et al. 2004; Shi and Yu 2005], contact angles [Wang et al. 2005], sand [Zhu and Bridson 2005], and two phase flow [Hong and Kim 2005].

There are a number of two-dimensional techniques for large bodies of water including the popular methods for deep water [Fournier and Reeves 1986; Peachey 1986; Mastin et al. 1987; Ts’o and Barsky 1987; Thon et al. 2000]. We refer the interested reader to the recent work of [Hinsinger et al. 2002] and course notes of [Tessendorf 2002] and [Baraff et al. 2003]. Of course, the deep water equations ignore bottom topography and do not resolve fully three-dimensional phenomena at the surface. Another interesting technique consists of solving the two-dimensional Navier-Stokes equations for the horizontal velocities, and then using the pressure to define a height field as in [Chen and Lobo 1994]. [Thon and Ghazanfarpour 2001] also solved the two-dimensional Navier-Stokes equations for the horizontal velocity in streams, but used a noise function for the vertical velocity. [Neyret and Prazelin 2001] proposed a simpler stream model using a two-dimensional Laplace equation for the bulk flow. Finally, a few authors have tackled large bodies of water with the three-dimensional Navier-Stokes equations, e.g. focusing on splash and foam in [Takahashi et al. 2003] and on breaking waves in [Mihalef et al. 2004].

3 Grid Structure

We simulate on a uniform two-dimensional horizontal grid of vertical columns that contain both uniform cells and tall cells that replace collections of uniform cells, e.g. see Figure 2. The tall and uniform cells can be arbitrary in each column, and in particular we

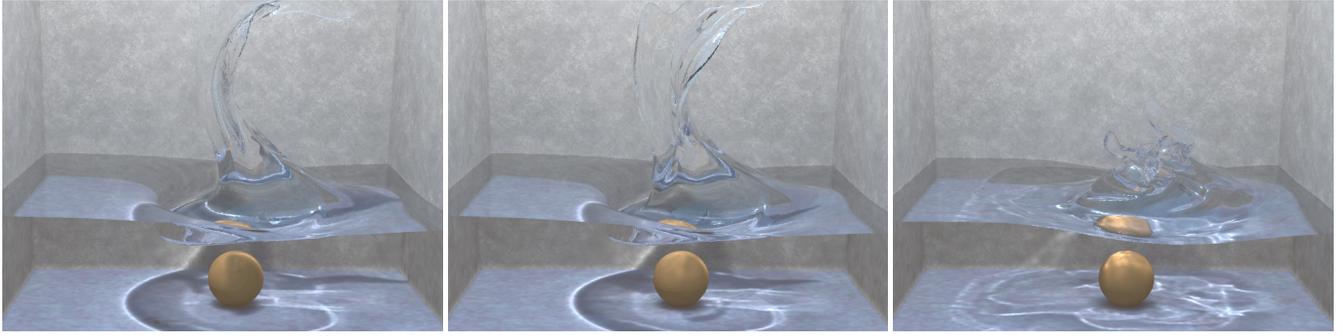


Figure 4: Three balls splashing into water (300×200 horizontal resolution). The fully refined case (left) and the $1/4$ refined case (middle) are quite similar. However, quite different results are obtained if one doesn't refine enough (right, $1/16$ refined).

split tall cells at the ground to match bottom topography (Figure 6). In a uniform MAC grid, scalars are stored at cell centers while velocity components are stored on their respective faces (Figure 3a). Level set values are only required in uniform cells near the interface, whereas pressure and velocity are needed everywhere. Tall cells contain two pressure values corresponding to the cell centers of the uppermost and bottommost uniform cells that the tall cell replaces (Figure 3b). Pressure values can be interpolated to the centers of the other uniform cells replaced by a tall cell using vertical linear interpolation.

Later, we will calculate horizontal pressure derivatives for every pressure sample (Figure 3c), and thus we co-locate horizontal velocities with them (Figure 3e). Figure 3e outlines the control volume around each *minimal* face noting that minimal faces between tall cells may contain two horizontal velocities (as shown in the figure). Between these two velocities, we interpolate horizontal velocities on the uniform faces replaced by this minimal face using linear interpolation consistent with the pressure and its derivatives. Vertical pressure derivatives and vertical velocities are shown in Figure 3d and 3f (also co-located) along with outlined control volumes. The linear pressure profile in the tall cell dictates that all uniform grid vertical faces replaced by the face at the center of this cell should have the same vertical velocity.

3.1 Refinement and Coarsening

When we change the structure of tall and uniform cells, the velocity needs to be interpolated to the new grid. The vertical velocities at the center of new tall cells are set to the average value of all the vertical faces replaced by it, whereas those for uniform cells are defined as their interpolated values. This conserves vertical momentum. For horizontal velocities, the difficulties occur when a new minimal face replaces more than one uniform face. For example, let $u(j_a), u(j_a + 1), \dots, u(j_b)$ be a sequence of velocity values on a minimal face. We desire bottom and top values u_a and u_b such that linear interpolation between them best approximates the sequence. The least squares error is

$$E = \frac{1}{2} \sum_{j=j_a}^{j_b} \left(\frac{j_b - j}{j_b - j_a} u_a + \frac{j - j_a}{j_b - j_a} u_b - u(j) \right)^2$$

Differentiating with respect to u_a and u_b and setting the derivatives to zero gives

$$\sum_{j=j_a}^{j_b} \frac{(j_b - j)^2}{j_b - j_a} u_a + \frac{(j_b - j)(j - j_a)}{j_b - j_a} u_b = \sum_{j=j_a}^{j_b} (j_b - j) u(j)$$

$$\sum_{j=j_a}^{j_b} \frac{(j_b - j)(j - j_a)}{j_b - j_a} u_a + \frac{(j - j_a)^2}{j_b - j_a} u_b = \sum_{j=j_a}^{j_b} (j - j_a) u(j)$$

Using the formulas for sums of linear and quadratic sequences, these equations simplify into

$$\frac{n+1}{6} \begin{pmatrix} 2n+1 & n-1 \\ n-1 & 2n+1 \end{pmatrix} \begin{pmatrix} u_a \\ u_b \end{pmatrix} = \sum_{j=j_a}^{j_b} \begin{pmatrix} j_b - j \\ j - j_a \end{pmatrix} u(j) \quad (1)$$

where $n = j_b - j_a$. If some subsequence of the $u(j)$'s come from a minimal face's linear profile, the same formulas can compute that portion of the right-hand side sum in constant time. This 2×2 system is trivially inverted to find u_a and u_b . As shown in [Houston et al. 2006], the entire process of building a new grid and transferring data between grids takes linear time.

We use a one grid cell band of uniform cells around moving objects, so that no special tall cell treatment is required for one-way or two-way coupling with solids (though we implemented only one-way coupling so far). In addition, we define an optical depth down to which we want to preserve as much detail as possible and use uniform cells within that distance from the interface. We emphasize that the optical depth required for a given problem does not shrink under refinement, rather it is a true length scale related to the three-dimensional physics we wish to capture.

3.2 Refinement Analysis and Comparison

The effects of varying the optical depth are illustrated in Figure 4. With a sufficient optical depth, satisfactory results are obtained (middle). Otherwise, we fail to resolve enough of the three-dimensional flow structure and obtain nonphysical results (right). These simulations used 4 processors in a 2×2 grid (see section 6), and took 250, 116, and 59 seconds per frame, respectively. Note that the fully refined splash (left) uses essentially the same method as [Houston et al. 2006].

Since the computational cost of a tall cell is independent of its height, all the tall cells together are equivalent in cost to a few extra layers of uniform cells regardless of the depth of the water. That is, if a certain optical depth of uniform cells is needed, then another layer of cells provides the extra computation needed for our approach. It is unlikely that another adaptive method such as an octree

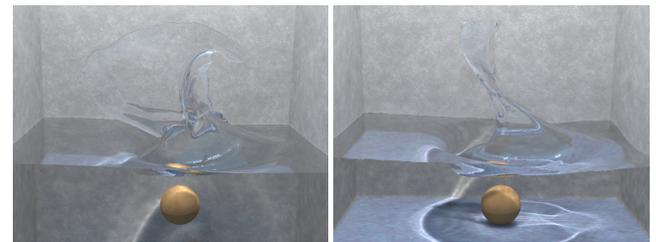


Figure 5: (Left) The same optical depth as Figure 4 (middle), but with twice the water depth. While this would be twice as expensive with a uniform grid, there is almost no cost increase using our approach. (Right) An octree simulation with two levels of coarsening away from the interface ($312 \times 208 \times 208$ effective resolution).

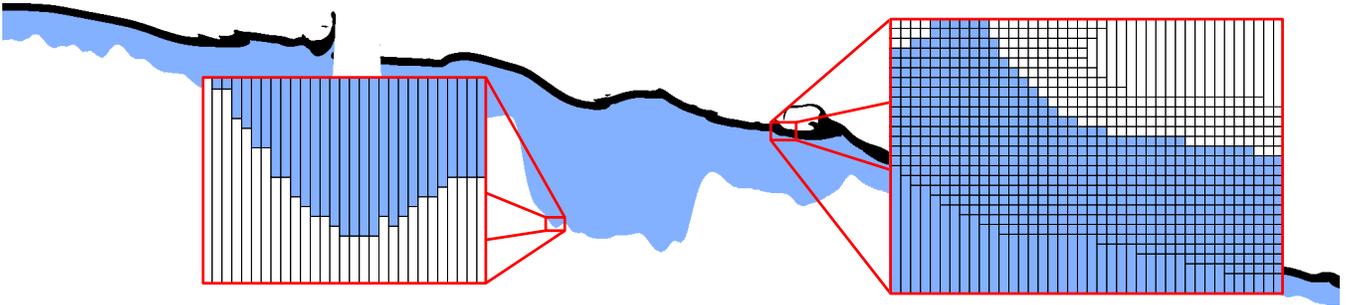


Figure 6: A cross-section of the grid from a river simulation showing tall cells used to represent bottom topography.

would make this layer more efficient. In particular, we can double the depth of the water without incurring the factor of two slowdown characteristic of uniform grids or previous RLE techniques (Figure 5 (left)). Moreover, the resulting splash is qualitatively different from the shallow simulation, which shows the importance of water depth on the bulk flow. The deeper simulation took 140 seconds a frame due to the increased surface area compared to Figure 4 (middle).

Figure 5 (right) shows a serial octree simulation of the same splash. Although Figure 4 (middle) takes the same amount of total computation (460 seconds per frame), our method is readily parallelized leading to faster turnaround and larger simulations. Excessive computational costs kept us from refining the octree to the same resolution *throughout* the entire optical depth region, and we instead were forced to coarsen away from the typical three grid cell band, which made the simulation significantly more viscous. This highlights an interesting aspect of our approach, in that we combine our tall cells with a *uniform* grid whereas it is probably better to combine them with an *adaptive* octree grid. In fact, this is straightforward to implement by placing individual octrees in each of our uniform cells just as was done in [Losasso et al. 2006].

4 Uniform Three-Dimensional Method

The inviscid, incompressible Navier-Stokes equations for the conservation of mass and momentum are

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p / \rho = \mathbf{g} \quad (2)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (3)$$

where $\mathbf{u} = (u, v, w)$ is the velocity, p is the pressure, ρ is the density, and \mathbf{g} is the acceleration due to gravity. We use a standard, uniform MAC grid fluid solver in the band of uniform cells, together with the particle level set method of [Enright et al. 2002]. Both the level set and velocity field are advected with semi-Lagrangian advection [Stam 1999] in this band. Since the level set is only defined near the interface, its semi-Lagrangian rays will interpolate from other uniform cells. In contrast, velocity is defined everywhere, and some semi-Lagrangian rays will ask for information from tall cells. These velocities will instead be updated with the same algorithm used to update velocities in tall cells (see section 5.1). Note that the pressure is solved for globally, coupling the uniform and tall cells together (see section 5.2). In situations where additional turbulence is desired, we use the vortex particle approach of [Selle et al. 2005]. Figures 1 and 7 show simulations with vortex particles seeded behind a boat to represent the turbulence generated by a propeller.

5 Adaptive Two-Dimensional Method

In this section, we address the advection of velocity on all the tall cells and on the uniform cells that were not updated with semi-Lagrangian advection because they were too close to the tall cells. We also address solving for the pressure on an arbitrary collection of tall and uniform cells. These discretizations will be constructed

by summing uniform finite volume discretizations over tall cells using linear or constant basis functions, and can be considered discrete finite volume/finite element methods.

5.1 Advection

We use first order accurate conservative upwinding for all tall cells and any uniform cells that are not updated with the semi-Lagrangian method. Since the *number* of cells that need to be updated with this method is a small subset of the total cells (even though it can be most of the volume), the efficiency of this approach is unimportant. Thus, we chose the conservative method to preserve momentum in these *very* tall cells that are likely to have large truncation errors. Moreover, conservative schemes are quite popular for the two-dimensional nonlinear shallow water equations, which have many similarities with our tall cells. We emphasize that conservation does not cure large truncation errors: it only makes the results more physically plausible.

We begin by putting the velocity terms into conservation form using $\nabla \cdot \mathbf{u} = 0$ to transform equation 2 into

$$\mathbf{u}_t + \nabla \cdot (\mathbf{u}\mathbf{u}^T) + \nabla p / \rho = \mathbf{g}.$$

Ignoring the pressure and force terms, we focus on the u component of the velocity

$$u_t + (u^2)_x + (vu)_y + (wu)_z = 0.$$

If we place a control volume around each face as shown by the dotted lines in Figure 3e, then this equation can be rewritten to indicate that the velocity is updated based on the fluxes across the control volume faces, i.e.

$$u^* = u^n + \frac{\Delta t}{\Delta x \Delta y \Delta z} \sum_{f=1}^6 \pm F_f \quad (4)$$

where the sign depends on which side of the control volume a flux is on. The fluxes are computed by averaging the velocities to the control volume faces, and then using these average velocities to decide on the upwind direction. The flux itself is constructed by multiplying the upwind component of the velocity we are advecting, u_{up} , with the average values for the other components v_{av} and w_{av} :

$$F_u = \Delta y \Delta z u_{up}^2, \quad F_v = \Delta x \Delta z v_{av} u_{up}, \quad F_w = \Delta x \Delta y w_{av} u_{up} \quad (5)$$

The v and w velocity components are treated similarly.

While this algorithm is straightforward on a uniform grid, a few modifications are required for tall cells. For motivation, we point out that we could refine the tall cells into a uniform grid, apply the uniform grid method just discussed, and then re-coarsen. Although this would be inefficient to implement directly, our approach achieves exactly this result in an efficient manner. To simulate this process of refining, advecting, and coarsening, we consider each pair of adjacent minimal faces one at a time, apply equations 5 and 4 to the entire velocity profile to produce a piecewise linear or quadratic u^* on each side, and use equation 1 to reduce u^* back to a constant or linear profile. The fact that reducing each flux contribution to u^* separately has the same result as treating them all

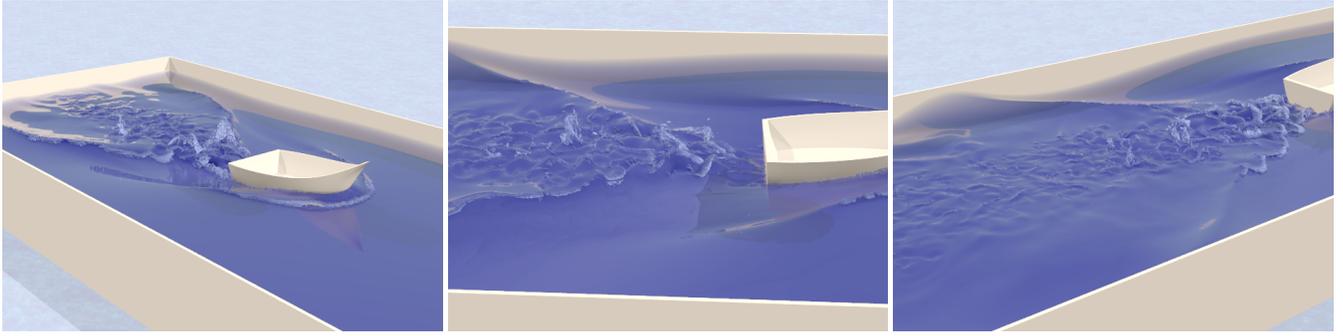


Figure 7: The boat moving along a straight path (1500 × 300 horizontal resolution).

at once follows from the linearity of equation 1. As shown in Figure 8, the structure of the flux profile differs depending on which component is being advected and in which direction.

For example, given two minimal u faces whose control volumes intersect along the x direction, the averaged velocity profile u_{av} will be linear. If u_{av} has constant sign, F_u has a single quadratic profile (u_{up} is linear) on the control volume face. If u_{av} changes sign, we split the control volume face into two sections with constant sign and handle each section separately. Advection of u along the other horizontal dimension w is similar.

Advecting u along v between distinct minimal u faces is simple since the intersection is a single uniform control volume face. However, we must also account for flux between the top and bottom samples of a linear profile. Averaging v to the middle control volume face produces a constant v_{av} at every virtual control volume face, so the flux profile F_v will be linear. The update of u^* can be simplified by noting that the net flux into every virtual uniform face except for the first and last is constant, since it is the difference of consecutive terms in a linear sequence.

Advection of the other horizontal velocity w is similar to that of u . Advection of v along u or w is the same as for u except that the flux profiles are linear instead of quadratic, and there is no need to consider flux inside middle v faces since they are constant.

First order upwinding requires a CFL condition on the time step for stability, namely $\Delta t \max(|u|/\Delta x + |v|/\Delta y + |w|/\Delta z) \leq 1$. If a larger time step is desired for the other parts of the algorithm, the upwinding phase of advection can be subcycled for little extra cost since it is only needed for a small fraction of the velocities. After advection, we refine/coarsen the grid. This is done before making the velocity field divergence free, because velocity interpolation does not preserve discrete incompressibility.

There are two issues with this advection method which we would like to resolve with future work. First, instabilities sometimes ap-

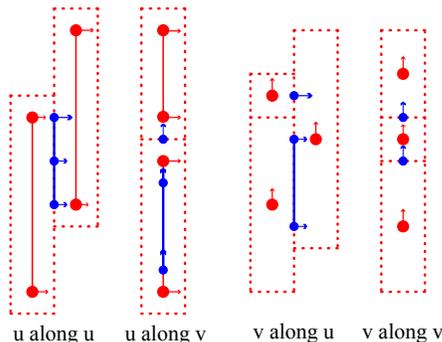


Figure 8: Advection fluxes (small blue circles) on minimal control volume faces between adjacent velocity control volumes.

pear near sharp changes in bottom topography such as the walls of the canyon in Figure 9. These can occur because the final u^* values are not affine combinations of the starting velocities, since averaging to control volume faces breaks discrete incompressibility. We avoided these instabilities by computing the total weight used for each u^* value, and dividing by this weight if u^* would be larger than an affine combination. Note that these weights can be computed by applying the same advection algorithm to the constant field 1 (i.e., substitute 1 for \mathbf{u}_{up} everywhere). This fix is far from satisfactory, especially since the lack of discrete incompressibility causes no problems for uniform grids. Second, while a first order method is sufficient for bulk motion, it would be interesting to generalize the ENO and WENO schemes typically used for shallow water to the case of tall cells to reduce numerical dissipation.

5.2 Laplace Equation

After advection, we solve for the pressure and make the velocities divergence free via

$$\begin{aligned} \nabla \cdot (\nabla p / \rho) &= \nabla \cdot \mathbf{u}^* / \Delta t \\ \mathbf{u}^{n+1} &= \mathbf{u}^* - \Delta t \nabla p / \rho \end{aligned}$$

for the entire collection of uniform and tall cells at once. Since variable density flows do not have approximately linear vertical pressure profiles, we assume that the density is spatially constant and can be moved to the right hand side, i.e. $\nabla \cdot \nabla p = \rho \nabla \cdot \mathbf{u}^* / \Delta t$.

The component of the discrete pressure gradient on a MAC grid face between two MAC grid cells p_1 and p_2 is $\partial p / \partial l = (p_2 - p_1) / \Delta l$ where $l = x, y, \text{ or } z$. This readily generalizes to arbitrary configurations of tall cells, since we determined the placement and structure of the velocities with pressure gradients in mind (Figure 3c,d). For the horizontal derivative, say in the x direction, consider two adjacent tall cells extending from j_1 to j_2 and j_3 to j_4 , respectively, intersecting in a minimal face from $j_a = \max(j_1, j_3)$ to $j_b = \min(j_2, j_4)$. If the corresponding pressures are p_1, p_2, p_3, p_4 , then the component of the discrete gradient on the face between them is a linear profile with values $(p_x)_a$ and $(p_x)_b$ given by interpolating pressures to j_a and j_b in each cell and applying standard central differencing, e.g.

$$(p_x)_a = \frac{1}{\Delta x} \left(\frac{j_4 - j_a}{j_4 - j_3} p_3 + \frac{j_a - j_3}{j_4 - j_3} p_4 - \frac{j_2 - j_a}{j_2 - j_1} p_1 - \frac{j_a - j_1}{j_2 - j_1} p_2 \right)$$

and similarly for $(p_x)_b$. The derivative between vertically adjacent pressure samples p_1 at j_1 and p_2 at j_2 is simply $p_y = (p_2 - p_1) / ((j_2 - j_1) \Delta y)$.

The discrete volume weighted divergence of a uniform cell can be written as

$$V(\nabla \cdot \mathbf{u}) = \sum_{f=1}^6 \pm u_f A_f$$

where the \pm sign is chosen based on which face p is being considered, and A_f is the area of the face. We generalize this equation to tall

cells by computing the flux into each virtual uniform cell in the tall cell, dividing each flux between the bottom and top samples in the cell, and adding up the contributions. *We divide the virtual uniform fluxes using the same fractions used to interpolate pressures to the given virtual uniform cell in order to ensure the symmetry of the final system.* For example, given a tall cell from j_1 to j_2 and a minimal face from j_a to j_b with velocities u_a and u_b , the total flux contribution from the face given to the upper portion of the cell is

$$\sum_{j=j_a}^{j_b} \frac{j-j_1}{j_2-j_1} \left(\frac{j_b-j}{j_b-j_a} u_a + \frac{j-j_a}{j_b-j_a} u_b \right) \Delta y \Delta z. \quad (6)$$

while

$$\sum_{j=j_a}^{j_b} \frac{j_2-j}{j_2-j_1} \left(\frac{j_b-j}{j_b-j_a} u_a + \frac{j-j_a}{j_b-j_a} u_b \right) \Delta y \Delta z.$$

is the contribution to the lower portion of the cell. The vertical flux contribution is always $v \Delta x \Delta z$.

We can now combine the discrete volume weighted divergence and gradient operators to discretize the Laplacian. For the horizontal direction, to get the contribution from the minimal face from j_a to j_b to the top portion of the cell, we substitute $(p_x)_a$ and $(p_x)_b$ for u_a and u_b in equation 6 and collapse the nested interpolation to obtain

$$\frac{\Delta y \Delta z}{\Delta x} \sum_{j=j_a}^{j_b} \frac{j-j_1}{j_2-j_1} \left(\frac{j_4-j}{j_4-j_3} p_3 + \frac{j-j_3}{j_4-j_3} p_4 - \frac{j_2-j}{j_2-j_1} p_1 - \frac{j-j_1}{j_2-j_1} p_2 \right)$$

and similarly

$$\frac{\Delta y \Delta z}{\Delta x} \sum_{j=j_a}^{j_b} \frac{j_2-j}{j_2-j_1} \left(\frac{j_4-j}{j_4-j_3} p_3 + \frac{j-j_3}{j_4-j_3} p_4 - \frac{j_2-j}{j_2-j_1} p_1 - \frac{j-j_1}{j_2-j_1} p_2 \right)$$

is the contribution to the bottom portion of the cell. Note that the coefficient of p_1 in the first equation (which is for p_2) is identical to the coefficient of p_2 in the second equation (which is for p_1). Since all symmetric pairs of contributions to the discretization matrix are identical, the final result is symmetric. The coefficients relating p_3 to p_4 are handled similarly. Moreover, the cross terms relating p_1 to p_3 , p_1 to p_4 , p_2 to p_3 and p_2 to p_4 are also similarly treated. For the vertical terms, we obtain $(\Delta x \Delta z / \Delta y)(p_2 - p_1) / (j_2 - j_1)$ once again resulting in identical contributions for symmetric terms. The resulting negative semi-definite linear system can be treated in the same manner as a typical MAC grid discretization, i.e. preconditioned conjugate gradients, incomplete Cholesky preconditioner, Neumann and Dirichlet boundary conditions anywhere in tall or uniform cells, etc.

6 Examples

We implemented our method in parallel using MPI by decomposing the horizontal domain into a two-dimensional grid and giving each processor one fixed rectangular piece of the domain. Before every stage of the fluid solve that requires neighboring data, we fill ghost cells on the sides of each processor from the boundary data of its neighboring processors. In the pressure solve, each processor constructs an incomplete Cholesky preconditioner for the portion of the grid that it owns, and participates in a global PCG solve with a block diagonal preconditioner formed from the incomplete Cholesky blocks on each processor. Extending the serial version of the algorithm to use MPI was relatively straightforward since the domain decomposition is applied only along uniform horizontal dimensions, and the use of tall cells for the bulk of the fluid significantly reduced the communication bandwidth required. All simulations were run on a cluster of 4 processor 2.6 GHz Opteron machines.

The boat simulations in Figures 1 and 7 used 16 processors in an 8×2 grid. The optical depth was $1/5$ the water depth and the cost was approximately 3 minutes a frame. For the boat examples we used vortex particles [Selle et al. 2005] and rendered removed negative particles as in [Guendelman et al. 2005]. Figure 9 shows a river

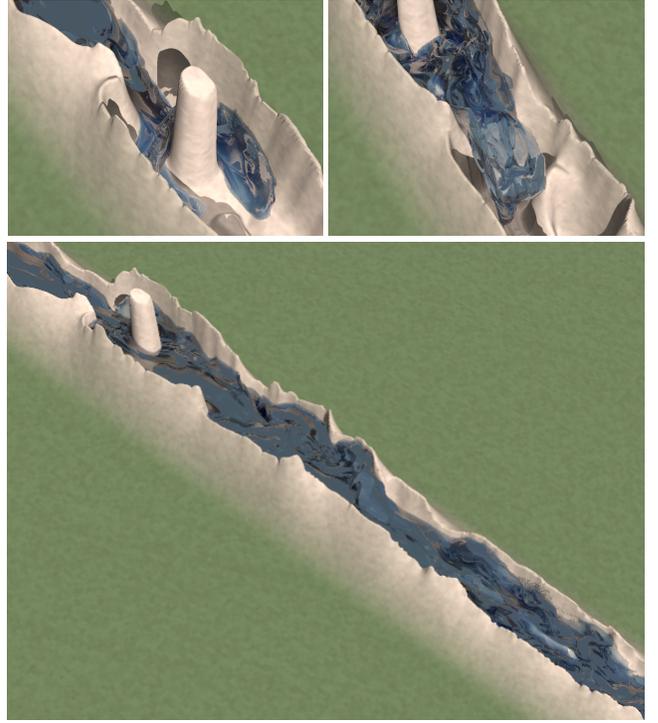


Figure 9: Simulation of a river filling a canyon (2000×200 horizontal resolution).

simulation with varying bottom topography using 20 processors in a 20×1 grid. The computational cost was approximately 25 minutes a frame, and the optical depth is illustrated in Figure 6. Memory consumption was not a significant issue in any of the simulations.

7 Conclusion

We have presented a novel method for the simulation of large bodies of water by combining two-dimensional and three-dimensional simulation techniques. The bulk of the water volume is represented with tall cells similar to a height field method, and a surface layer of water is simulated with a state of the art, fully three-dimensional Navier-Stokes free surface solver. This algorithm works well for capturing detailed surface motion and for representing detailed bottom topography. Our general philosophy is to use the best available method in the surface layer where we expect interesting detail, and to coarsen the mesh away from the interface for efficiency.

Like shallow water, our method only admits computational gains for flows heavily dominated by gravity, where a large portion of the water is in near vertical equilibrium against the ground. In general, we believe that the best approach for any simulation is to start with a uniform grid that captures the interesting flow features, coarsen vertically using our method in areas where linear pressure profiles are sufficient, and then refine uniform cells using an octree in areas where more resolution is desirable. Finally, we note that the effects of bottom topography on the surface diminish with increasing depth, and in the deep water limit the effects are negligible. However, we are interested in problems in the shallow water regime, where tall cells are almost exclusively used.

Acknowledgements

Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ONR N00014-03-1-0071, ONR N00014-02-1-0720, ONR N00014-05-1-0479 (for a SUN computing cluster), ARO DAAD19-03-1-0331, NSF IIS-0326388, NSF

ITR-0205671, NSF ITR-0121288, NSF ACI-0323866 and NIH U54-GM072970. G.I. was supported in part by an NSF Graduate Fellowship.

References

- ADALSTEINSSON, D., AND SETHIAN, J. 1995. A fast level set method for propagating interfaces. *J. Comput. Phys.* 118, 269–277.
- BARAFF, D., WITKIN, A., KASS, M., AND ANDERSON, J. 2003. Physically based modeling (a little fluid dynamics for graphics). In *SIGGRAPH Course Notes*, ACM.
- BAXTER, W., AND LIN, M. 2004. Haptic interaction with fluid media. In *Proc. of Graph. Interface*, 81–88.
- BAXTER, W., LIU, Y., AND LIN, M. 2004. A viscous paint model for interactive applications. In *Proc. of Comput. Anim. and Social Agents*, vol. 15, 433–441.
- BAXTER, W., WENDT, J., AND LIN, M. 2004. Impasto: A realistic, interactive model for paint. In *Proc. of Non-Photorealistic Anim. and Rendering*, 45–56.
- BREEN, D., FEDKIW, R., MUSETH, K., OSHER, S., SAPIRO, G., AND WHITAKER, R. 2004. Level sets and PDE methods for computer graphics. In *SIGGRAPH Course Notes*, ACM.
- BRIDSON, R. 2003. *Computational Aspects of Dynamic Surfaces*. PhD thesis, Stanford University.
- CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (SIGGRAPH Proc.)* 23, 377–384.
- CHEN, J., AND LOBO, N. 1994. Toward interactive-rate simulation of fluids with moving obstacles using the navier-stokes equations. *Comput. Graph. and Image Processing* 57, 107–116.
- CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. *Comput. Graph. (SIGGRAPH Proc.)*, 303–312.
- ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.)* 21, 3, 736–744.
- FEDKIW, R., STAM, J., AND JENSEN, H. 2001. Visual simulation of smoke. In *Proc. of ACM SIGGRAPH 2001*, 15–22.
- FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proc. of ACM SIGGRAPH 2001*, 23–30.
- FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models and Image Processing* 58, 471–483.
- FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. In *Comput. Graph. Int.*, 178–188.
- FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *Proc. of SIGGRAPH 97*, 181–188.
- FOURNIER, A., AND REEVES, W. T. 1986. A simple model of ocean waves. In *Comput. Graph. (Proc. of SIGGRAPH 86)*, vol. 20, 75–84.
- GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graph. (SIGGRAPH Proc.)* 23, 463–467.
- GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH Proc.)* 24, 3, 973–981.
- HINSINGER, D., NEYRET, F., AND CANI, M.-P. 2002. Interactive animation of ocean waves. In *ACM SIGGRAPH Symp. on Comput. Anim.*, 161–166.
- HONG, J.-M., AND KIM, C.-H. 2005. Discontinuous fluids. *ACM Trans. Graph. (SIGGRAPH Proc.)* 24, 3, 915–919.
- HOUSTON, B., WIEBE, M., AND BATTY, C. 2004. RLE sparse level sets. In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.
- HOUSTON, B., NIELSEN, M., BATTY, C., NILSSON, O., AND MUSETH, K. 2005. Gigantic deformable surfaces. In *SIGGRAPH 2005 Sketches & Applications*, ACM Press.
- HOUSTON, B., NIELSEN, M., BATTY, C., NILSSON, O., AND MUSETH, K. 2006. Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Trans. Graph.* 25, 1, 1–24.
- IVERSEN, J., AND SAKAGUCHI, R. 2004. Growing up with fluid simulation on “The Day After Tomorrow”. In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.
- KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. In *Comput. Graph. (Proc. of SIGGRAPH 90)*, vol. 24, 49–57.
- LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)* 23, 457–462.
- LOSASSO, F., FEDKIW, R., AND OSHER, S. 2006. Spatially Adaptive Techniques for Level Set Methods and Incompressible Flow. *Computers and Fluids (in press)*.
- MASTIN, G., WATTERBERG, P., AND MAREDA, J. 1987. Fourier synthesis of ocean scenes. *IEEE Comput. Graph. Appl.* 7, 3, 16–23.
- MCMANAMA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 449–456.
- MIHALEF, V., METAXAS, D., AND SUSSMAN, M. 2004. Animation and control of breaking waves. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 315–324.
- NEYRET, F., AND PRAIZELIN, N. 2001. Phenomenological simulation of brooks. In *Comput. Anim. and Sim. '01*, Proc. Eurographics Wrkshp., 53–64.
- NIELSEN, M., AND MUSETH, K. 2005. Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *Accepted to SIAM J. Scientific Comput.*
- O'BRIEN, J. F., AND HODGINS, J. K. 1995. Dynamic simulation of splashing fluids. In *Comput. Anim.* '95, 198–205.
- PEACHEY, D. R. 1986. Modeling waves and surf. In *Comput. Graph. (Proc. of SIGGRAPH 86)*, vol. 20, 65–74.
- PENG, D., MERRIMAN, B., OSHER, S., ZHAO, H., AND KANG, M. 1999. A PDE-based fast local level set method. *J. Comput. Phys.* 155, 410–438.
- SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH Proc.)* 24, 3, 910–914.
- SHI, L., AND YU, Y. 2005. Taming liquids for rapidly changing targets. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 229–236.
- STAM, J. 1999. Stable fluids. In *Proc. of SIGGRAPH 99*, 121–128.
- TAKAHASHI, T., FUJII, H., KUNIMATSU, A., HIWADA, K., SAITO, T., TANAKA, K., AND UEKI, H. 2003. Realistic animation of fluid with splash and foam. *Comp. Graph. Forum (Eurographics Proc.)* 22, 3, 391–400.
- TESSENDORF, J. 2002. Simulating Ocean Water. In *SIGGRAPH 2002 Course Notes #9 (Simulating Nature: Realistic and Interactive Techniques)*, ACM Press.
- THON, S., AND GHAZANFARPOUR, D. 2001. A semi-physical model of running waters. *Comput. Graph. Forum (Proc. Eurographics)* 19, 53–59.
- THON, S., DISCHLER, J.-M., AND GHAZANFARPOUR, D. 2000. Ocean waves synthesis using a spectrum-based turbulence function. In *Comput. Graph. Int.*, 65–74.
- TS'O, P. Y., AND BARSKY, B. A. 1987. Modeling and rendering waves: Wave-tracing using beta-splines and reflective and refractive texture mapping. *ACM Trans. Graph.* 6, 3, 191–214.
- WANG, H., MUCHA, P., AND TURK, G. 2005. Water drops on surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.)* 24, 3, 921–929.
- WHITAKER, R. T. 1998. A level-set approach to 3d reconstruction from range data. *Int. J. Comput. Vision* 29, 3, 203–231.
- WIEBE, M., AND HOUSTON, B. 2004. The tar monster: Creating a character with fluid simulation. In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.
- ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. (SIGGRAPH Proc.)* 24, 3, 965–971.