An Efficient and Stable Algorithm for Animating Water Waves

Anita W. Tam

Department of Computer Science, University of Toronto

Toronto, Ontario, Canada, M5S 3G4

tam@cs.toronto.edu

Michiel van de Panne

Department of Computer Science, University of Toronto

Toronto, Ontario, Canada, M5S 3G4

van@cs.toronto.edu

Abstract

Water motions can be realistically captured by physically-based fluid models, which take advantage of the huge amount of effort invested in analyzing fluid motion in areas such as physics and fluid dynamics. We begin by presenting a survey on fluid simulation models that are based on fluid dynamics equations, from the most comprehensive Navier-Stokes equations, to the simple wave equation. For each model, we discuss the assumptions involved, its applications and limitations, and its computational complexity. We then present a model, based on the two-dimensional shallow water equations, for simulating ocean dynamics. The equations are integrated by a novel numerical method — the implicit semi-Lagrangian integration scheme — which allows large timesteps while maintaining stability, and is described in detail in this paper. Gentle wave motions, the superposition of waves, drifting objects, and obstacles and boundaries of various shapes can be efficiently simulated with our model. The model has been shown to be accurate, stable, efficient and parallelizable.

Keywords : fluid dynamics, physically-based, shallow-water, semi-Lagrangian

1 Introduction

Water motion is a complex phenomenon of which the interesting behaviour has yet to be fully captured in interactive computer simulation. This work focuses on developing and analyzing an efficient and stable algorithm for animating waves based on a set of two-dimensional shallow water equations. By taking advantage of the large amount of effort already invested in analyzing water motion in areas such as physics and fluid dynamics, a physically-based fluid model is capable of producing realistic wave motions.

The first contribution of this paper is a delineation of fluid dynamics equations, on which fluid animation models have been based. These equations vary greatly in complexity, with the most comprehensive model being the Navier-Stokes equations, which capture the turbulent or stable behaviour of fluid with arbitrary viscosity in three dimensions. At the other end of the spectrum we find the simple wave equation, which describes the sinusoidal propagation of a simple wave.

In section 3, we outline a physically-based model for animating water waves that is based on the two-dimensional shallow water equations, which fall in the middle of the complexity spectrum of fluid models, and capture gentle ocean waves. In our model, the equations are integrated by the implicit semi-Lagrangian integration scheme, which allows large timesteps while maintaining stability, and will be described in detail in this paper. We will show how our model can be used to animate water waves and objects drifting on the water, and to incorporate obstacles and boundaries of various shapes. In section 4, we analyze our algorithm, compare it with other models, and conclude that our algorithm is both stable and efficient.

2 Survey of Fluid Dynamics Models

A vast amount of effort has been invested in the field of computational fluid dynamics to design and solve models that describe fluid motions. These models vary in their levels of complexity and comprehensiveness, and some are better suited for animating a certain phenomenon than others. In this section, we summarize a number of fluid dynamics models that have been adopted in simulating motions of incompressible fluids (i.e. liquids), point out the scope of applicability and limitations of each model, and compare their computational complexities.

The Navier-Stokes equations, notably the most comprehensive of all fluid models, describe the motion of a fluid particle at an arbitrary location in the fluid field at any instant of time, and are derived from the well-known Newton's second law of motion. In three dimension, for an incompressible fluid, the equations take the following form:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = g_x - \frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = g_y - \frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) \tag{2}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = g_z - \frac{1}{\rho}\frac{\partial p}{\partial z} + \nu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) \tag{3}$$

where u, v and z are fluid velocity components in the x-, y- and z-directions, respectively, p denotes the pressure, g denotes the gravitational constant, ρ is the density, assumed constant, and ν is the kinematic viscosity of the fluid. The Navier-Stokes equations are usually applied in conjunction with the continuity equation, which states the law of conservation of mass:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{4}$$

Simulation models based on the Navier-Stokes equations, such as that proposed by Foster and Metaxas [5], are highly realistic, since they capture three-dimensional dynamics of fluid with arbitrary viscosity. Interesting phenomena such as a jet of water splashing into a tank, or large ocean waves crashing into a shallow cove can be simulated with a high level of realism. The downside of such models is that, since the equations are three-dimensional, the algorithm takes $\mathcal{O}(N^3)$ time to run, where N is the number of grid points in one dimension. Thus, interactive rate is difficult to achieve for these models with the current state of technology.

Chen et. al. [1, 2] achieved interactive-rate simulation by eliminating the vertical dependence, and solve the resulting two-dimensional Navier-Stokes equations. The fluid surface is then elevated by computing its height from the pressure field. They were able to animate fluids of various viscosities by varying the Reynolds number. However, the adoptation of an explicit time integration method leads to potential instabilities unless small timesteps are chosen, which in turn increases the computational cost. Moreover, their single-layered two-dimensional approach cannot easily model liquid flows in a container with curved boundary, or around an object with cross-sectional area that varies with depth.

By assuming zero viscosity, and considering only two-dimensional motions, a yet simpler set of fluid equations can be derived — the shallow water equations, which describe flows of thin layers of fluids:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + g\frac{\partial h}{\partial x} = 0 \tag{5}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial h}{\partial y} = 0 \tag{6}$$

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}[u(h-b)] + \frac{\partial}{\partial y}[v(h-b)] = 0 \tag{7}$$

In the above equations, h is the height field and b is the height of the ground, as shown in Fig. 1. On the thin boundary of a liquid, the height field and the pressure field are related by $p = g\rho h$. Equations (5) and (6) are derived from Newton's second law of motion, whereas equation (7) is the continuity equation.

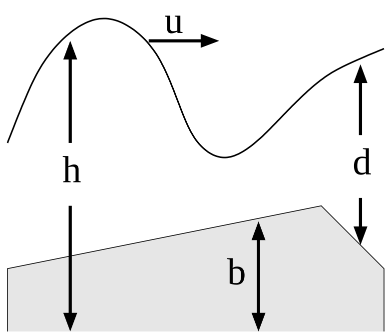

Figure 1: A diagram of a one-dimensional wave.

The shallow water equations are often adopted in oceanographic or atmospheric applications for large-scale global modeling, where the Coriolis force, induced by the rotation of the Earth, is included. For animation purposes, however, the Coriolis force is considered negligible and has been excluded in the above equations, since it is orders of magnitude smaller than other quantities in the equations and is therefore insignificant except in engineering applications. The shallow water equations are limited to the description of two-dimensional inviscid flows, which means that fluids with high viscosity cannot be modeled, nor can the fluids splash or break. Nonetheless, these equations serve well in their purpose of modeling gentle ocean waves, and in fact our model is based on the shallow water equations.

In order for the simulation to be stable even for reasonably large timesteps, an implicit time integration scheme should be used. However, the nonlinear advection terms in the equations (the second and third terms) render the system nontrivial to solve. Kass and Miller [13] proposed a solution to this problem by assuming that water speed varied slowly in space, and solved in realtime the following linearized shallow water equations:

$$\frac{\partial u}{\partial t} + g\frac{\partial h}{\partial x} = 0 \tag{8}$$

$$\frac{\partial v}{\partial t} + g\frac{\partial h}{\partial y} = 0 \tag{9}$$

$$\frac{\partial h}{\partial t} + d\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \tag{10}$$

where $d = h - b$ is the water depth. In section 4, we compare and analyze the accuracy and efficiency of our model and [13].

As one of the first attempts at modeling water waves, Fournier and Reeves [7] presented a simple model, based on two-dimensional wave equations, for animating breaking waves near a sloping beach. Peachey [16] presented a similar approach. However, their models lack predictability as wave motions need to be determined a priori, and user-introduced perturbations may not be handled easily. This means that it may be difficult to build interactive applications from their models, and motions such as dropping of an object into water cannot be simulated.

3 The Model

3.1 The Shallow Water Equations

It is the goal of this paper to construct a physically-based model for the realistic and interactive-rate simulation of ocean dynamics. We choose to base our model on the nonlinear shallow water equations, with which predictions can be made of the ways the water height (h) changes with time. The coupled equations are integrated in time using the *implicit semi-Lagrangian* method, which we will describe in section 3.2. Unlike the explicit Euler integration scheme, the implicit semi-Lagrangian method integrates along particle trajectories and is thus stable for large timesteps. Moreover, the nonlinear advection terms can be handled easily. Since in our algorithm the water speed is computed for each timestep, simulations can also be done for objects drifting on the water surface, as we will demonstrate in section 4.

3.2 The Implicit Semi-Lagrangian Time Integration Method

The semi-Lagrangian method can be viewed as a hybrid between the Eulerian and the Lagrangian approaches [4, 21]. In an Eulerian-based advection scheme, the observer stays at a fixed geographical point as the world, or the fluid, evolves around him. This scheme retains the regularity of the mesh as the observer stays fixed, but requires small timesteps in order to maintain stability. In a Lagrangian-based scheme, on the other hand, the observer watches the world evolve while traveling with a fluid particle. This technique is less restricted by stability requirements and allows larger timesteps. However, since the fluid particles move with time, the initially regularly-spaced set of fluid particles usually become irregularly-spaced as the system evolves. The semi-Lagrangian advection scheme attempts to combine the advantages of both schemes — the regularity of the Eulerian scheme and the enhanced stability of the Lagrangian scheme — by integrating along particle trajectories while evaluating target functions at mesh points at every timestep.

To show how the shallow water equations can be integrated by the semi-Lagrangian scheme, we first consider the one-dimensional shallow water equations:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + g\frac{\partial h}{\partial x} = 0 \tag{11}$$

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}[u(h-b)] = 0 \tag{12}$$

which can be rewritten in Lagrangian form as follows:

$$\frac{du}{dt} + g\frac{\partial h}{\partial x} = 0 \tag{13}$$

$$\frac{dh}{dt} - u\frac{\partial b}{\partial x} + d\frac{\partial u}{\partial x} = 0 \tag{14}$$

where $d = h - b$ and the Lagrangian derivative is defined as

$$\frac{d}{dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x}$$

in which

$$\frac{dx}{dt} = u(x,t) \tag{15}$$

As shown in Fig. 2, the Lagrangian derivatives are approximated along the trajectories. Imagine a fluid particle that travels along its trajectory and arrives at position x_i at time t_{n+1}. Then according to the semi-Lagrangian method, the derivatives are computed from its positions at t_{n+1}, which is x_i, and at t_n, which is called the *departure point* and is marked as x^* in Fig. 2:

$$\frac{du}{dt} = \frac{u^{n+1} - \tilde{u}^n}{\Delta t}, \qquad \frac{dh}{dt} = \frac{h^{n+1} - \tilde{h}^n}{\Delta t}. \tag{16}$$

In the above equations, u^{n+1} and h^{n+1} are evaluated at grid points at time-level t_{n+1}, while $\tilde{u}^n$ and $\tilde{h}^n$ are evaluated at departure points at t_n. Details for calculating the trajectories and computing the departure points can be found in appendix A.

We make the assumption that water depth (d) is approximately constant within the time interval (t_n, t_{n+1}). By adopting approximations (16), we discretize equations

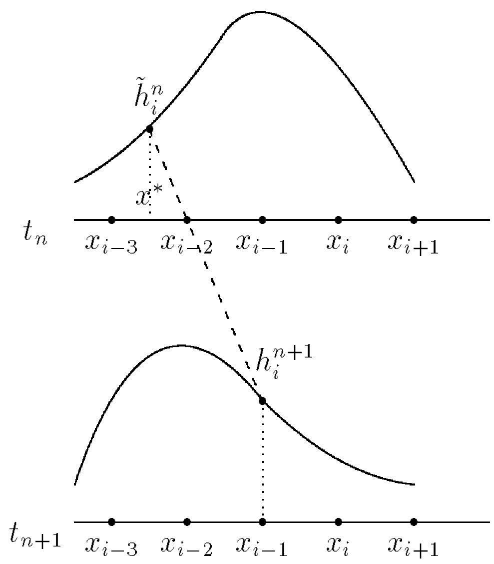

Figure 2: With the semi-Lagrangian scheme, the Lagrangian derivative is approximated along particle trajectories.

(13) and (14) in time implicitly to yield the following equation:

$$\frac{u^{n+1} - \tilde{u}^n}{\Delta t} + g\frac{\partial h}{\partial x}^{n+1} = 0 \tag{17}$$

$$\frac{h^{n+1} - \tilde{h}^n}{\Delta t} - u^{n+1}\frac{\partial b}{\partial x} + d^n\frac{\partial u}{\partial x}^{n+1} = 0 \tag{18}$$

The height of the ground, b, is assumed to be time-independent, so there is no time-level superscript associated with it.

The method described above is first order, in both time and space, and implicit. The latter implies that large timesteps can be taken without loss of stability. Equations (17) and (18) can be solved by taking the partial derivative of (17) with respect to x, and then using the resulting equation and (17) to eliminate the $\partial u^{n+1}/\partial x$ and u^{n+1} terms from (18) to yield a Helmholtz equation in h^{n+1} only:

$$h^{n+1} + \Delta t^2 g\frac{\partial b}{\partial x}\frac{\partial h}{\partial x}^{n+1} - \Delta t^2 g d^n\frac{\partial^2 h}{\partial x^2}^{n+1} = \tilde{h}^n + \Delta t\tilde{u}^n\frac{\partial b}{\partial x} - \Delta t d^n\frac{\partial \tilde{u}}{\partial x}^n \tag{19}$$

Spatially discretizing the above equation using centered difference yields the following tridiagonal system, of which the solution is the new height field:

$$
\begin{aligned}
& h_i^{n+1} + \Delta t^2 g \left(\frac{b_{i+1} - b_{i-1}}{2\Delta x}\right)\left(\frac{h_{i+1}^{n+1} - h_{i-1}^{n+1}}{2\Delta x}\right) - \Delta t^2 g d_i^n \left(\frac{h_{i-1}^{n+1} - 2h_i^{n+1} + h_{i+1}^{n+1}}{\Delta x^2}\right) \\
= \; & \tilde{h}_i^n + \Delta t \tilde{u}_i^n \left(\frac{b_{i+1} - b_{i-1}}{2\Delta x}\right) - \Delta t d_i^n \left(\frac{\tilde{u}_{i+1}^n - \tilde{u}_{i-1}^n}{2\Delta x}\right) \qquad (20)
\end{aligned}
$$

3.2.1 The Two-Dimensional Case

In two dimensions, the time discretized shallow water equations take the following form:

$$
\begin{aligned}
\frac{u^{n+1} - \tilde{u}^n}{\Delta t} + g \frac{\partial h}{\partial x}^{n+1} &= 0 \qquad (21) \\
\frac{v^{n+1} - \tilde{v}^n}{\Delta t} + g \frac{\partial h}{\partial y}^{n+1} &= 0 \qquad (22) \\
\frac{h^{n+1} - \tilde{h}^n}{\Delta t} - \left(u^{n+1} \frac{\partial b}{\partial x} + v^{n+1} \frac{\partial b}{\partial y}\right) + d^n \left(\frac{\partial u}{\partial x}^{n+1} + \frac{\partial v}{\partial y}^{n+1}\right) &= 0 \qquad (23)
\end{aligned}
$$

As in the one-dimensional case, we take the appropriate spatial derivatives of (21) and (22) and eliminate the divergence term from (23). The resulting Helmholtz equation in h is

$$
\begin{aligned}
& h^{n+1} + \Delta t^2 g \left(\frac{\partial b}{\partial x}\frac{\partial h}{\partial x}^{n+1} + \frac{\partial b}{\partial y}\frac{\partial h}{\partial y}^{n+1}\right) - \Delta t^2 g d^n \left(\frac{\partial^2 h}{\partial x^2}^{n+1} + \frac{\partial^2 h}{\partial y^2}^{n+1}\right) \\
= \; & \tilde{h}^n + \Delta t \left(\tilde{u}^n \frac{\partial b}{\partial x} + \tilde{v}^n \frac{\partial b}{\partial y}\right) - \Delta t d^n \left(\frac{\partial \tilde{u}}{\partial x}^n + \frac{\partial \tilde{v}}{\partial y}^n\right) \qquad (24)
\end{aligned}
$$

The above equation is then spatially discretized to become

$$
\begin{aligned}
& h_{i,j}^{n+1} + \Delta t^2 g \left(\frac{b_{i+1,j} - b_{i-1,j}}{2\Delta x}\frac{h_{i+1,j}^{n+1} - h_{i-1,j}^{n+1}}{2\Delta x} + \frac{b_{i,j+1} - b_{i,j-1}}{2\Delta y}\frac{h_{i,j+1}^{n+1} - h_{i,j-1}^{n+1}}{2\Delta y}\right) \\
& -\Delta t^2 g d_{i,j}^n \left(\frac{h_{i-1,j}^{n+1} - 2h_{i,j}^{n+1} + h_{i+1,j}^{n+1}}{\Delta x^2} + \frac{h_{i,j-1}^{n+1} - 2h_{i,j}^{n+1} + h_{i,j+1}^{n+1}}{\Delta y^2}\right) \\
= \; & \tilde{h}_{i,j}^n + \Delta t \left(\tilde{u}_{i,j}^n \frac{b_{i+1,j} - b_{i-1,j}}{2\Delta x} + \tilde{v}_{i,j}^n \frac{b_{i,j+1} - b_{i,j-1}}{2\Delta y}\right)
\end{aligned}
$$

$$-\Delta t d_{i,j}^{n}\left(\frac{\tilde{u}_{i+1,j}^{n}-\tilde{u}_{i-1,j}^{n}}{2\Delta x}+\frac{\tilde{v}_{i,j+1}^{n}-\tilde{v}_{i,j-1}^{n}}{2\Delta y}\right) \tag{25}$$

which is solved with the conjugate gradient method [11]. Water speed u^{n+1} and v^{n+1} can be updated by back-substituting h^{n+1} into (21) and (22).

Algorithm 1 The wave simulation algorithm is summarized below:

```
Initialize h(0), u(0), v(0) and b.
loop for all timesteps t_{n+1}:
    Compute departure points using equation (31).
    Compute h, u and v at departure points using equation (32).
    Solve equation (25) for h^{n+1}.
    Update u^{n+1} and v^{n+1} with equations (21) and (22).
end loop
```

4 Applications and Analysis

4.1 Examples

Various types of fluid movements can be animated. In the first example, illustrated in Fig. 3, we implemented a realtime three-dimensional simulation of water waves in a rectangular pool, using fixed and non-reflective boundary conditions. Since our model is physically based, we need only to supply it with a set of initial conditions (i.e. the shape of the water surface at the start of the simulation) for the waves to evolve naturally according to the shallow water equations.

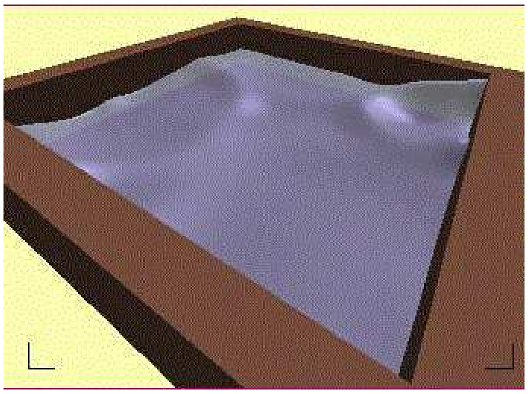

Figure 3: Simulation of water waves in a rectangular pool.

Objects drifting with the water can also be animated with ease, as the water speed is computed explicitly in our algorithm. Let (p_x^n, p_y^n) denote the position of the floating object at time t_n. Then its position at t_{n+1} is given by

$$p_x^{n+1} = p_x^n + u^n(p_x^n, p_y^n)\Delta t \tag{26}$$

$$p_y^{n+1} = p_y^n + v^n(p_x^n, p_y^n)\Delta t \tag{27}$$

If (p_x^n, p_y^n) is not one of the grid points, then u^n and v^n can be approximated with linear interpolation as in (32).

We also include examples to demonstrate our algorithm's ability to handle non-trivial boundary conditions. In one example, shown in Fig. 4, blocks of various shapes are placed in the pool; in another example, water waves are animated in a triangular pool. In our implementation, we constrain the perpendicular component of the water velocity to be zero at the walls. Specifically, the grid points along a wall which runs North-South should have zero East-West velocities, and vice versa. For concave corners, such as the four corners of a rectangular pool, all the velocities are set to be zero. For a convex corner, such as the corner of a rectangular block surrounded by water, the component of the velocity incident to the corner at a 45-degree angle is set to be zero.

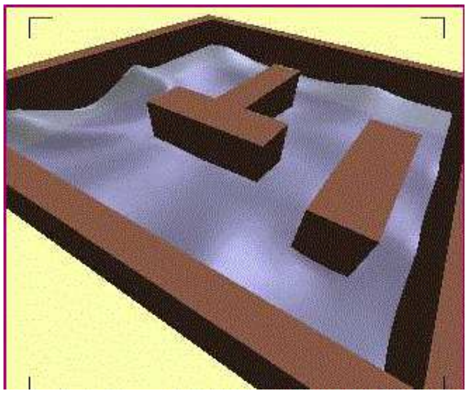

Figure 4: Examples of wave animation with non-trivial boundary conditions.

4.2 Solution Stability and Accuracy

Explicit time integration methods, because of their simplicity, were used in many implementations of fluid models [1, 2, 5]. However, nothing comes without a price. The major disadvantage of explicit integration methods is that severe restriction is imposed on the timestep size. In other words, unless a very small timestep is chosen, the numerical solution may diverge exponentially from the true solution.

By choosing an implicit integration method, we ensure that stability is not restricted by the magnitude of the gradient terms (the second terms in (13) and (14)). By adopting the semi-Lagrangian approach, we also ensure that the timestep size is not limited by the Courant-Fredrichs-Lewy (CFL) condition [4]. Our algorithm is stable as long as the departure points are estimated with sufficient accuracy. In other words, as long as the estimated departure points ($x - \alpha^n(x)$) fall between the two grid points (x_{m-1} and x_m) used in the linear interpolation (32), the integration remains stable. We will avoid a lengthy mathematical analysis here, but it suffices to say that as long as the product of the timestep (Δt) and wind sheer ($\max(|u_x|, |u_y|, |v_x|, |v_y|)$) is bounded by some constant, numerical stability is ensured [17].

The stability of our algorithm is compared experimentally to an explicit method. Various initial conditions were tested, including examples with a water drop, an upside-down gauss hill, and a moving wave front, and for each case, the largest timestep possible before instability occurred was measured. A numerical solution is considered unstable when, over a long period of simulation, it fails to closely resemble the expected solution, obtained using a small timestep and a fine spatial grid, or, in our algorithm, the conjugate gradient solver does not converge. In all of our test cases, the maximum timestep obtained for the implicit semi-Lagrangian method is up to ten times larger than that for the explicit method. This relaxation in timestep restriction allows much fewer timesteps to be taken for the same length of simulation, thereby reducing the total computation cost tremendously.

Our model adopts the complete shallow water equations so that the animations may be natural and realistic. Consequently, our solution should be more accurate in the physical sense than that of, say, Kass and Miller [13], who based their model on a linearized version of the shallow water equations. Our numerical solution is first order in both time and space, and, as pointed out earlier in this section, is relatively stable even for large time steps. Thus it may be concluded that our solution is accurate both physically and numerically.

4.3 Efficiency

In this section we give a simple analysis of the numerical complexity of our algorithm. The majority of the computation effort is spent solving the Helmholtz equation (25). Let N and the number of grid points in one dimension, and k be the number of iterations in conjugate gradient, employed in solving (25). By taking advantage of the sparsity of the matrices, the conjugate gradient method has a complexity of $\mathcal{O}(N^2k)$ [19]. In all our simulations, one or two iterations sufficed to attain the level of accuracy desired. Therefore, our algorithm has an overall complexity of $\mathcal{O}(N^2)$.

We compare the efficiency of our method to that proposed by Kass and Miller [13]. Note that, unlike our model, they assumed that water speed varied slowly in space, and solved the linearized shallow water equations without the nonlinear advection terms, (8) to (10). The hyperbolic equation for h, after eliminating u and v from the system, takes the form:

$$\frac{\partial^2 h}{\partial t^2} = gd\left(\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2}\right) \tag{28}$$

The above equation was solved with the Alternating Direction Implicit (ADI) method. However, equations (19) and (20) in [13] appear to be erroneous, since the dependence of h on x and y is largely uncoupled. The correct ADI algorithm should be:

1. For $k = 1, 2, \ldots,$ until convergence, do:

2. Solve the following equation in x-direction:

$$\frac{\partial^2 h^{n+\frac{1}{2},[k]}}{\partial t^2} - gd^n \frac{\partial^2 h^{n+\frac{1}{2},[k]}}{\partial x^2} = gd^n \frac{\partial^2 h^{n,[k-1]}}{\partial y^2} \tag{29}$$

3. Using newly computed $h^{n+\frac{1}{2},[k]}$, solve the following equation in y-direction:

$$\frac{\partial^2 h^{n+1,[k]}}{\partial t^2} - gd^n \frac{\partial^2 h^{n+1,[k]}}{\partial y^2} = gd^n \frac{\partial^2 h^{n+\frac{1}{2},[k]}}{\partial x^2} \tag{30}$$

In our study equations (29) and (30) are used in place of (19) and (20) in [13]. The linear systems that are solved in steps 2 and 3 are derived in appendix B. When tested on a workstation with a Sparc Ultra 4 processor, using the same test cases as those in the stability test in section 4.2, the computational portion of our algorithm, though solving more complicated equations, was approximately four times faster than that of Kass and Miller. This speedup may be attributed to the faster convergence rate of conjugate gradient used in our algorithm over the ADI method chosen by Kass and Miller, and that only one linear system is solved in our algorithm, while two are solved in that of Kass and Miller. The ADI method also has the disadvantage of being non-scalable on a parallel machine because of the global data exchanges involved between the two ADI steps. Global communication introduces undesirable communication delays especially when the number of processors is large. The conjugate gradient method, on the other hand, is known to be highly scalable and can therefore be easily implemented on a parallel machine [9].

5 Discussion

We have presented a physically-based model to animate water waves. The model is capable of producing realistic wave motions, and has been shown to be accurate, efficient and stable even with large timesteps. Furthermore, compared to the work of Kass and Miller [13], our algorithm has been shown to be superior in both accuracy (since it is based on the complete shallow water equations) and efficiency (four-time speedup). With this speedup, more detailed ocean scenes may be simulated in real-time.

The limitations of our model lie in that the shallow water equations are restricted to the description of inviscid flows of thin layers of fluids. In other words, phenomena such as three-dimensional flows, fluids with high viscosity and breaking waves cannot be modeled. Nonetheless, our model serves well in its purpose of modeling gentle ocean waves and drifting objects.

To address the above limitations, and in part motivated by the efficiency of our integration technique, we plan to develop a more sophisticated fluid model based on the Navier-Stokes equations in subsequent work. Though the Navier-Stokes equations capture all fluid motions, there are few realtime applications that are based on these equations due to their computational complexity. Current techniques either fail to achieve interactive rate, or are unstable over a long simulation with large timesteps [1, 2, 5]. It is therefore our goal to build a fluid simulation model, based on the Navier-Stokes equations, that is capable of animating fluids with differing viscosities, and turbulent motions in three dimensions.

A Trajectory Calculations

We describe in detail how the departure points are computed in section 3.2. In equations (17) and (18), $\tilde{u}^n$ and $\tilde{h}^n$ are evaluated at *departure points* $(x - \alpha^n(x))$ at time-level t_n in Fig. 5. α^n denotes the distance that a fluid particle travels in the time interval t_n to t_{n+1}, and is computed by integrating (15) backward in time. With a first order approximation,

$$\alpha^n = \Delta t u^n \tag{31}$$

Note that the departure points are often off-mesh points. This means that values for $\tilde{u}^n$ and $\tilde{h}^n$ may not be known, in which case they are approximated with linear interpolation. Let x_m denote the m-th grid point: $x_m \equiv m\Delta x$. Suppose $x_{m-1} < x - \alpha^n(x) < x_m$, then

$$\tilde{u}^n(x - \alpha^n(x)) = \frac{(x - \alpha^n(x) - x_{m-1})u^n(x_m) + (x_m - x + \alpha^n(x))u^n(x_{m-1})}{\Delta x} \tag{32}$$

A similar formula can be applied to the height field to obtain its values at departure

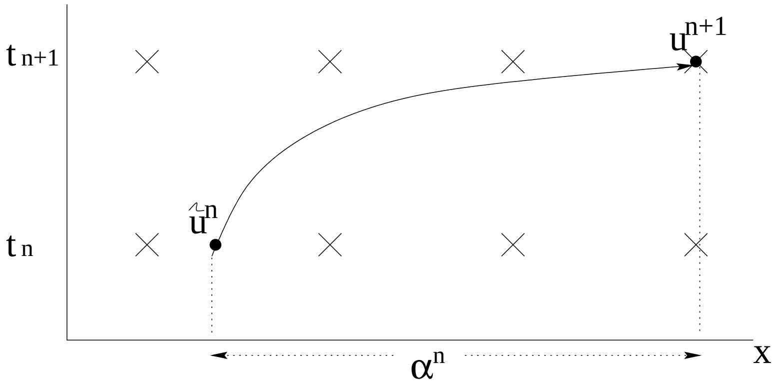

Figure 5: An illustration of the semi-Lagrangian time integration method.

points.

B Details for ADI iterations

We will now derive the linear systems that are solved in steps 2 and 3 of the ADI iterations outlined in section 4.3. In the formulation of Kass and Miller [13], a staggered system is used in which the height field h is computed on mesh points that are staggered with respect to those of the wave speed u and v. (Refer to Fig. 1 in [13] for an illustration) After spatial discretization using a first-order finite difference method, the shallow water equations (8) to (10) become

$$\begin{aligned}
\frac{\partial h_{i,j}}{\partial t} &= \left(\frac{d_{i-1,j}+d_{i,j}}{2\Delta x}\right)u_{i-1,j} - \left(\frac{d_{i,j}+d_{i+1,j}}{2\Delta x}\right)u_{i,j} \\
&\quad + \left(\frac{d_{i,j-1}+d_{i,j}}{2\Delta y}\right)v_{i,j-1} - \left(\frac{d_{i,j}+d_{i,j+1}}{2\Delta y}\right)v_{i,j} && (33) \\
\frac{\partial u_{i,j}}{\partial t} &= -g\left(\frac{h_{i+1,j}-h_{i,j}}{\Delta x}\right) && (34) \\
\frac{\partial v_{i,j}}{\partial t} &= -g\left(\frac{h_{i,j+1}-h_{i,j}}{\Delta y}\right) && (35)
\end{aligned}$$

Assuming that the water depth d is constant over each time step, we take the time derivative of (33), eliminate from the resulting equation the u and v dependence using (34) and (36), and obtain the following partial differential equations in h:

$$\begin{aligned}\frac{\partial^2 h_{i,j}}{\partial t^2} &= -g\left(\frac{d_{i-1,j}+d_{i,j}}{2\Delta x^2}\right)(h_{i,j}-h_{i-1,j}) + g\left(\frac{d_{i,j}+d_{i+1,j}}{2\Delta x^2}\right)(h_{i+1,j}-h_{i,j}) \\ &\quad -g\left(\frac{d_{i,j-1}+d_{i,j}}{2\Delta y^2}\right)(h_{i,j}-h_{i,j-1}) + g\left(\frac{d_{i,j}+d_{i,j+1}}{2\Delta y^2}\right)(h_{i,j+1}-h_{i,j})\end{aligned} \tag{36}$$

The goal of the first ADI sub-iteration (29), is to march forward half a timestep using information from previous time-levels n and $n-1$. The first and second time derivatives of the height field h are approximated as follows:

$$\ddot{h}^{n+\frac{1}{2}} = \frac{\dot{h}^{n+\frac{1}{2}} - \dot{h}^n}{\frac{\Delta t}{2}} \tag{37}$$

$$\dot{h}^{n+\frac{1}{2}} = \frac{h^{n+\frac{1}{2}} - h^n}{\frac{\Delta t}{2}} \tag{38}$$

$$\dot{h}^{n} = \frac{h^{n} - h^{n-1}}{\Delta t} \tag{39}$$

Combining equation (36), (37), (38) and (39), we obtain the equation for the first ADI sub-iteration:

$$\begin{aligned} h_{i,j}^{n+\frac{1}{2}} &+ g\left(\frac{\Delta t}{2}\right)^2\left(\frac{d_{i-1,j}+d_{i,j}}{2\Delta x^2}\right)(h_{i,j}^{n+\frac{1}{2}} - h_{i-1,j}^{n+\frac{1}{2}}) \\ &- g\left(\frac{\Delta t}{2}\right)^2\left(\frac{d_{i,j}+d_{i+1,j}}{2\Delta x^2}\right)(h_{i+1,j}^{n+\frac{1}{2}} - h_{i,j}^{n+\frac{1}{2}}) \\ &= \frac{3}{2}h_{i,j}^n - \frac{1}{2}h_{i,j}^{n-1} - g\left(\frac{\Delta t}{2}\right)^2\left(\frac{d_{i,j-1}+d_{i,j}}{2\Delta y^2}\right)(h_{i,j}^n - h_{i,j-1}^n) \\ &\quad + g\left(\frac{\Delta t}{2}\right)^2\left(\frac{d_{i,j}+d_{i,j+1}}{2\Delta y^2}\right)(h_{i,j+1}^n - h_{i,j}^n) \end{aligned} \tag{40}$$

In the second sub-iteration, we solve for h^{n+1} using its values at time-levels t_n and $t_{n+\frac{1}{2}}$. The principle equation for this sub-iteration can be derived similarly:

$$h_{i,j}^{n+1} + g\left(\frac{\Delta t}{2}\right)^2\left(\frac{d_{i,j-1}+d_{i,j}}{2\Delta y^2}\right)(h_{i,j}^{n+1} - h_{i,j-1}^{n+1})$$

$$
\begin{aligned}
&- \quad g\left(\frac{\Delta t}{2}\right)^2\left(\frac{d_{i,j}+d_{i,j+1}}{2\Delta y^2}\right)(h_{i,j+1}^{n+1}-h_{i,j}^{n+1}) \\
&= \quad 2h_{i,j}^{n+\frac{1}{2}}-h_{i,j}^{n}-g\left(\frac{\Delta t}{2}\right)^2\left(\frac{d_{i-1,j}+d_{i,j}}{2\Delta x^2}\right)(h_{i,j}^{n+\frac{1}{2}}-h_{i-1,j}^{n+\frac{1}{2}}) \\
&\quad\quad +g\left(\frac{\Delta t}{2}\right)^2\left(\frac{d_{i,j}+d_{i+1,j}}{2\Delta x^2}\right)(h_{i+1,j}^{n+\frac{1}{2}}-h_{i,j}^{n+\frac{1}{2}}) \qquad (41)
\end{aligned}
$$

In our experiments, two ADI iterations for each timestep suffice to produce the desired level of accuracy.

References

[1] Jim X. Chen and Niels Da Vitoria Lobo. Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. *Graphical Models and Image Processing*, 57(2):107–116, March 1995.

[2] Jim X. Chen, Niels Da Vitoria Lobo, Charles E. Hughes, and J. Michael Moshell. Real-time fluid simulation in a dynamic virtual environment. *IEEE Computer Graphics and Applications*, pages 52–61, May-June 1997.

[3] R. Courant, E. Isaacson, and M. Rees. On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences. *Communication on Pure and Applied Mathematics*, 5:243–255, 1952.

[4] Dale R. Durran. *Numerical Method for Wave Equations in Geophysical Fluid Mechanics*. Springer, 1998.

[5] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphics Interface*, pages 204–211, 1996.

[6] Nick Foster and Dimitri Metaxas. Modeling the motion of a hot, turbulent gas. *ACM SIGGRAPH*, pages 181–188, 1997.

[7] Alain Fournier and William T. Reeves. A simple model of ocean waves. *Computer Graphics*, 20(4):75–84, August 1986.

[8] Michael E. Goss. A real time particle system for display of ship wakes. *IEEE Computer Graphics and Applications*, pages 30–35, May 1990.

[9] Anshul Gupta. Performance and scalability of preconditioned conjuate gradient methods on parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, 6:455–469, May 1995.

[10] George J. Haltiner and Roger T. Williams. *Numerical Prediction and Dynamic Meteorology*. John Wiley and Sons, 1980.

[11] M. R. Hestenes and E. Stiefel. Methods of conjugate gradient for solving linear systems. *Journal Research National Bureau of Standards*, 45:409–436, 1952.

[12] James R. Holton. *An Introduction to Dynamic Meteorology.* San Diego Academic Press, 1992.

[13] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. *Computer Graphics*, 24(4):49–55, August 1990.

[14] Dvaid Mould and Yee-Hong Yang. Modeling water for computer graphics. *Computer and Graphics*, 21(6):801–814, 1997.

[15] Beny Neta. Analysis of finite elements and finite differences for shallow water equations: A review. *Mathematics and Computers in Simulation*, 34:141–161, 1992.

[16] Darwyn R. Peachey. Modeling waves and surf. *Computer Graphics*, 20(4):65–74, August 1986.

[17] J. Pudykiewicz, R. Benoit, and A. Staniforth. Preliminary results from a partial LRTAP model used on an existing meteorological forecast model. *Atmospheric Oceans*, 23:267–303, 1985.

[18] André Robert. A semi-Lagrangian, semi-implicit numerical integration scheme for the primitive meteorological equations. *Atmosphere-Ocean*, 19:35–46, 1981.

[19] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific Statistical Computation*, 7:856–869, July 1986.

[20] Joe Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. *Computer Graphics Proceedings*, pages 369–376, 1993.

[21] Andrew Staniforth and Jean Côté. Semi-Lagrangian integration schemes for atmospheric models – a review. *Monthly Weather Review*, 119:2206–2223, September 1991.

[22] S. Thomas, J. Côté, and A. Staniforth. A semi-implicit semi-Lagrangian shallow-water model for massively parallel processors. Manuscript, 1993.

[23] Jakub Wejchert and David Haumann. Animation aerodynamics. *Computer Graphics*, 25(4):19–22, July 1991.