The Visual Computer

A numerically efficient and stable algorithm for animating water waves

Anita T. Layton†,
Michiel van de Panne

Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, M5S 3G4
E-mail: layton@ucar.edu, van@cs.toronto.edu

Published online: 24 January 2002

Water motion can be realistically captured by physically based fluid models. 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. We then present a model that is based on the two-dimensional shallow water equations. The equations are integrated by a novel numerical method – the implicit semi-Lagrangian integration scheme – which allows large timesteps while maintaining stability, and which 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 this model.

Key words: Fluid dynamics – Physically based model – Shallow water equations – Semi-Lagrangian method – Implicit integration

†*Present address:* Department of Mathematics, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA

1 Introduction

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

We begin this paper with a delineation of fluid dynamics equations, upon 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 Sect. 3, we outline a physically based model for animating water waves that is based on the two-dimensional shallow water equations. This formulation falls in the middle of the complexity spectrum of fluid models and captures gentle ocean waves. In this model, the equations are integrated by the implicit semi-Lagrangian integration scheme, which allows large timesteps while maintaining stability, and which will be described in detail in this paper. We will show how the model can be used to animate water waves and objects drifting on the water, and how to incorporate obstacles and boundaries of various shapes. Boundary conditions are handled by setting the perpendicular components of the velocity to zero.

In Sect. 4, we analyse our algorithm, compare it with other models, and demonstrate that our algorithm is both stable and computationally efficient with a complexity of $\mathcal{O}(N^2)$, where N is the number of grid subintervals in one dimension.

2 Survey of fluid dynamics models

A vast amount of effort has been invested in the field of computational fluid dynamics to create 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 proposed for simulating the motions of incom-

The Visual Computer (2002) 18:41–53
Digital Object Identifier (DOI) 10.1007/s003710100131

pressible fluids (i.e. liquids), point out the scope of applicability and limitations of each model, and compare their computational complexity.

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 Newton's second law of motion. In three dimensions, the equations for an incompressible fluid take the following form:

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

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

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

where u, v and w are fluid velocity components in the x-, y- and z-directions, respectively; p denotes the pressure; g denotes the gravitational constant; ρ denotes the density, assumed constant; and ν denotes 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:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \quad (4)$$

Simulation models based on the Navier–Stokes equations, such as that proposed by Foster and Metaxas (1996), are highly realistic, since they capture the 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 subintervals in one dimension. Thus, interactive rates are difficult to achieve at present for these models.

Stam (1999) proposed a stable model, based on the Navier–Stokes equations, that produces complex fluid-like flows. By solving the equations with the method of characteristics, and treating the diffusion terms implicitly, large timesteps may be used. However, the problems of simulating fluids with free boundaries (such as water) and with obstacles were not addressed.

Chen and his co-workers achieved interactive-rate simulation by eliminating the vertical dependence and solving the resulting two-dimensional Navier–Stokes equations (Chen and Lobo 1995; Chen et al. 1997). The fluid surface is then elevated by computing its height from the pressure field. They animated fluids of various viscosities by varying the Reynolds number. However, the adoption 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.

Also worth mentioning are commercial fluid simulators that are based on the Navier–Stokes equations. One example is *RealWave* by Next Limit S.L. RealWave tracks the propagation of gravitational waves on a mesh, and simulates liquid surfaces like the sea. It solves the simplified Navier–Stokes equations, with a simulated viscosity factor, using an explicit numerical integration scheme. Another example is *Digital NatureTools* by Areté Entertainment. In this model, the ghost fluid method is used to capture a contact discontinuity in a multiphase incompressible flow (e.g. an air–water interface), and the level set function is evolved over time (Kang et al. 2000).

Wave motions, including breaking and overturning waves, have also been investigated and effectively modelled. Longuet-Higgins and Cokelet (1975, 1978) proposed a numerical method for following the time history of space-periodic irrotational surface waves using the velocity potential of marked fluid particles at the free surface. At each timestep, an integral equation is solved for the new normal component of velocity. The number of independent variables in the computation is $\mathcal{O}(N^2)$. In this formulation, both viscosity and surface tension are ignored.

By assuming zero viscosity and considering only two-dimensional motions, an even simpler set of fluid equations can be derived – the shallow wa-

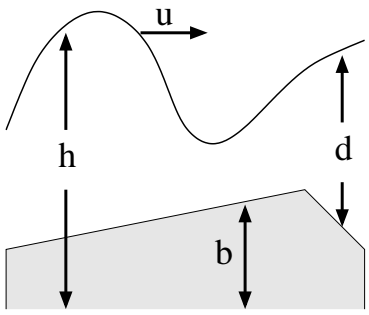

Fig. 1. One-dimensional shallow water wave

ter equations, which describe flows of thin layers of fluids (Haltiner and Williams 1980):

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

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

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

Here 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.

The shallow water equations are often adopted in oceanographic or atmospheric applications for large-scale global modelling, 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 modelled, and that the fluids cannot splash or break. Nonetheless, these equations adequately model a reasonably large class of fluid motions, and we thus choose to base our animation model on the shallow water equations.

In order for the simulation to be stable for 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 (1990) proposed a solution to this problem by assuming that water speed varied slowly in space, and solved in real time 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 Sect. 4 we compare and analyse the efficiency of our model and that of Kass and Miller.

Khan (1994) proposed a kinematic model for animating wakes created by a ship moving along an arbitrary course. Instead of using the dynamic solution of a free-surface problem, the model superimposes circular waves emanating from points along the ship's path to determine the wake profile.

As one of the first attempts at modelling water waves, Fournier and Reeves (1986) presented a simple model, based on two-dimensional wave equations, for animating breaking waves near a sloping beach. Peachey (1986) 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 those resulting from an object dropping into water cannot be simulated.

3 The model

3.1 The shallow water equations

The goal of this paper is 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 is described

in Sect. 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.

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 (Durran 1998; Staniforth and Côté 1991). In an Eulerian 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 scheme, on the other hand, the observer watches the world evolve while travelling with a fluid particle. This technique is less restricted by stability requirements and allows larger timesteps. However, since the fluid particles, initially regularly spaced, move with time, they 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.

The semi-Lagrangian method was first introduced in the atmospheric community by Robert (1981). In most engineering applications, the semi-Lagrangian method is used in conjunction with the *semi-implicit* approach, where the non-advection terms are averaged in time in order to achieve second-order accuracy. We choose to adopt the relatively simple implicit approach, which does not involve any time-averaging but produces only first-order solutions, which should be accurate enough for graphics applications.

When applied to an advection equation, the semi-Lagrangian method is equivalent to the classical method of characteristics (Courant et al. 1952). However, the semi-Lagrangian method retains its simplicity and practical utility in more complicated applications where the characteristics curves may deviate from the fluid particle trajectories, since the evolution of the flow continues to be computed following fluid particle trajectories. The method of characteristics, on the other hand, becomes unwieldy or impossible in more general problems where the evolution of the flow along characteristics curves may be more complicated or characteristics curves may not even be defined. For the distinction between the semi-Lagrangian method and the method of characteristics, we refer interested readers to Durran (1998).

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

$$\frac{\mathrm{d}u}{\mathrm{d}t} + g\frac{\partial h}{\partial x} = 0, \tag{13}$$

$$\frac{\mathrm{d}h}{\mathrm{d}t} - u\frac{\partial b}{\partial x} + d\frac{\partial u}{\partial x} = 0, \tag{14}$$

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

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

in which

$$\frac{\mathrm{d}x}{\mathrm{d}t} = u(x,t). \tag{15}$$

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 $\tilde{x}_i^n$ in Fig. 2. Let α_i^n be the displacement of a fluid particle in the time interval from t_n to t_{n+1}, ending at the downstream point x_i. For an arbitrary function $\psi(x,t)$, let $\tilde{\psi}^n$ denote the corresponding upstream function for the time interval from t_n to t_{n+1}: $\alpha_i^n = x_i - \tilde{x}_i^n$. That is, $\tilde{\psi}^n(x_i) \equiv \psi(x_i - \alpha_i^n, t_n)$. Thus, the Lagrangian derivatives are approximated as

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

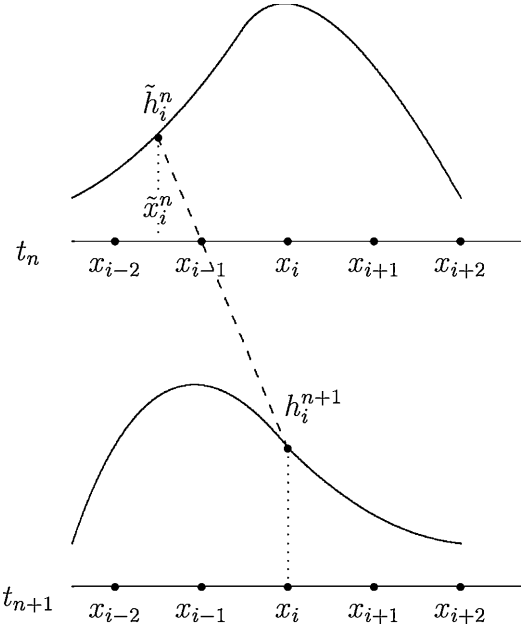

Fig. 2. With the semi-Lagrangian scheme, the Lagrangian derivative is approximated along particle trajectories

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

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

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

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 centred difference yields the following tridiagonal system, of which the solution is the new height field:

$$h_i^{n+1}+\Delta t^2 g\left(\frac{b_{i+1}-b_{i-1}}{2\Delta x}\right)\left(\frac{h_{i+1}^{n+1}-h_{i-1}^{n+1}}{2\Delta x}\right) -\Delta t^2 g d_i^n\left(\frac{h_{i-1}^{n+1}-2h_i^{n+1}+h_{i+1}^{n+1}}{\Delta x^2}\right) =\tilde{h}_i^n+\Delta t\tilde{u}_i^n\left(\frac{b_{i+1}-b_{i-1}}{2\Delta x}\right)-\Delta t d_i^n\left(\frac{\tilde{u}_{i+1}^n-\tilde{u}_{i-1}^n}{2\Delta x}\right). \tag{20}$$

3.2.1 The two-dimensional case

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

$$\frac{u^{n+1}-\tilde{u}^n}{\Delta t}+g\frac{\partial h}{\partial x}^{n+1}=0, \tag{21}$$

$$\frac{v^{n+1}-\tilde{v}^n}{\Delta t}+g\frac{\partial h}{\partial y}^{n+1}=0, \tag{22}$$

$$\frac{h^{n+1}-\tilde{h}^n}{\Delta t}-\left(u^{n+1}\frac{\partial b}{\partial x}+v^{n+1}\frac{\partial b}{\partial y}\right) +d^n\left(\frac{\partial u}{\partial x}^{n+1}+\frac{\partial v}{\partial y}^{n+1}\right)=0. \tag{23}$$

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

$$h^{n+1}+\Delta t^2 g\left(\frac{\partial b}{\partial x}\frac{\partial h}{\partial x}^{n+1}+\frac{\partial b}{\partial y}\frac{\partial h}{\partial y}^{n+1}\right) -\Delta t^2 g d^n\left(\frac{\partial^2 h}{\partial x^2}^{n+1}+\frac{\partial^2 h}{\partial y^2}^{n+1}\right) =\tilde{h}^n+\Delta t\left(\tilde{u}^n\frac{\partial b}{\partial x}+\tilde{v}^n\frac{\partial b}{\partial y}\right)-\Delta t d^n\left(\frac{\partial \tilde{u}}{\partial x}^{n}+\frac{\partial \tilde{v}}{\partial y}^{n}\right). \tag{24}$$

This is then spatially discretized to become

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

which is solved with the conjugate gradient method (Hestenes and Stiefel 1952). Water speed u^{n+1} and v^{n+1} can be updated by back-substituting h^{n+1} into (21) and (22).

The wave simulation algorithm is summarized below:

1. Initialize h^0, u^0, v^0 and b.
2. For each timestep t_{n+1} do:
   i. Compute departure points using equation (31).
   ii. Compute h, u and v at departure points using equation (32).
   iii. Solve equation (25) for h^{n+1}.
   iv. Update u^{n+1} and v^{n+1} with equations (21) and (22).

4 Applications and analysis

4.1 Examples

Various types of fluid movements can be animated. In the first example, we implemented a real-time simulation of water waves in a rectangular pool, using fixed and non-reflective boundary conditions;[1] the frames, shown in Fig. 3, are taken at 1 s intervals. The animations are rendered interactively using OpenGL, although off-line raytracing could also be applied to correctly model refraction effects as well as caustics cast on the bottom of the pool. Since our model is physically based, we need only 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.

All simulations were performed on a 600 MHz Pentium III processor with 128 MB of RAM. A square grid measuring 100 m on each edge and with a depth (b) of 10 m was used. A timestep of Δt =0.1 s was used and the simulations were run for 5 min. For a 40×40 and a 80×80 grid, the simulations took 12.5 s and 55.7 s, respectively, assuming no display. Run times with interactively rendered display depend on the display hardware as well as the display timestep.

Objects drifting with the water can also be animated, 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

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

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

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). In this model, the interaction of objects with the water is "one-way" in that the objects are affected by the motions of the water, but not vice versa. Thus, physical effects such as wave diffraction and radiation are not captured. This simplification is valid as long as the characteristic lengths of the objects are small compared to the wavelength.

We also include examples to demonstrate the ability of our algorithm 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 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° angle is set to zero. Homogeneous Neumann conditions are imposed on the water height at the boundaries; that is, we assume that the water height does not change across the boundaries.

[1] Reflected waves are implicitly modelled by the boundary conditions, as in the real world.

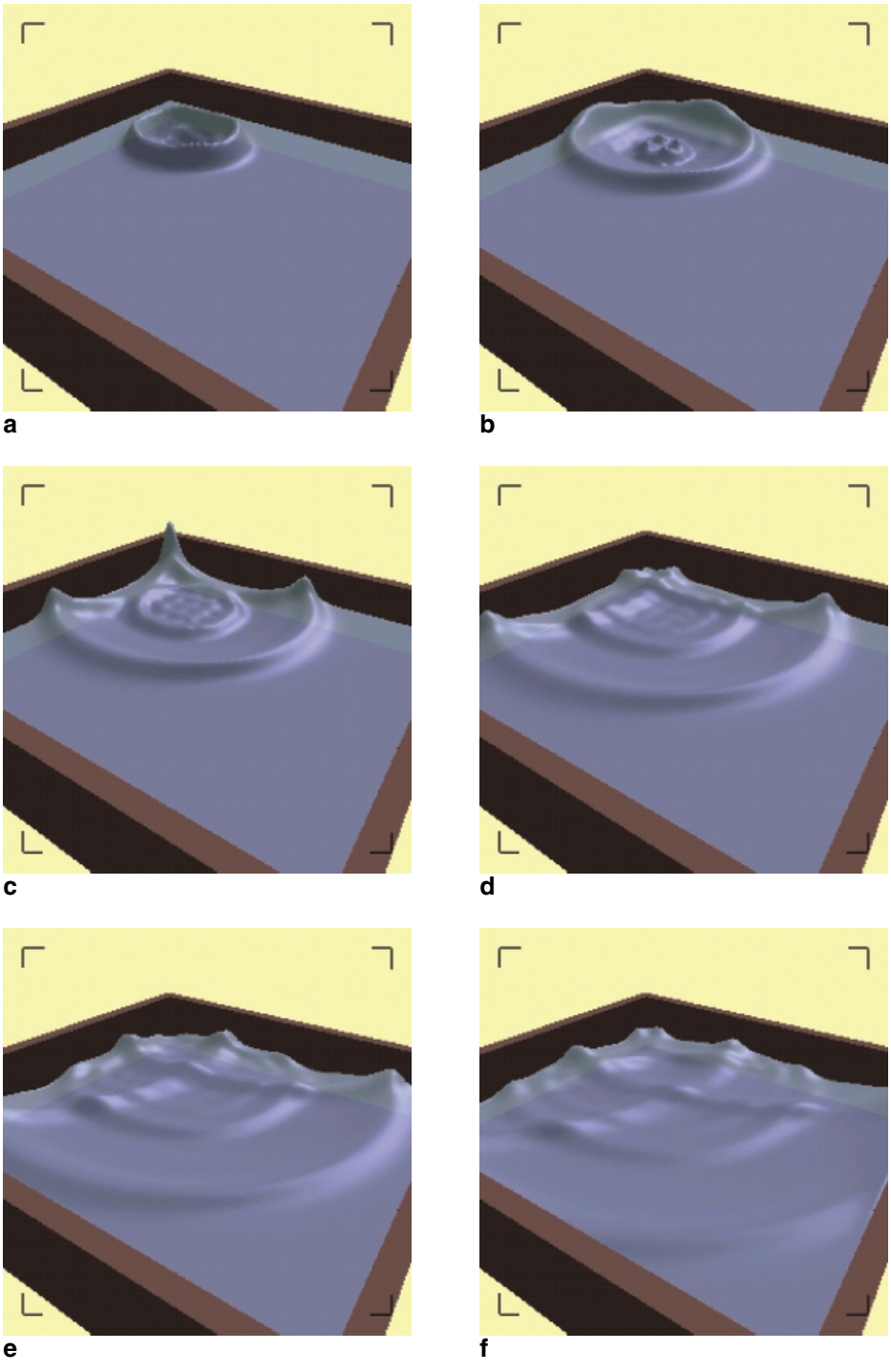

Fig. 3a–f. Simulation of water waves in a rectangular pool. **a** First frame; **b** second frame; **c** third frame; **d** fourth frame; **e** fifth frame; **f** sixth frame

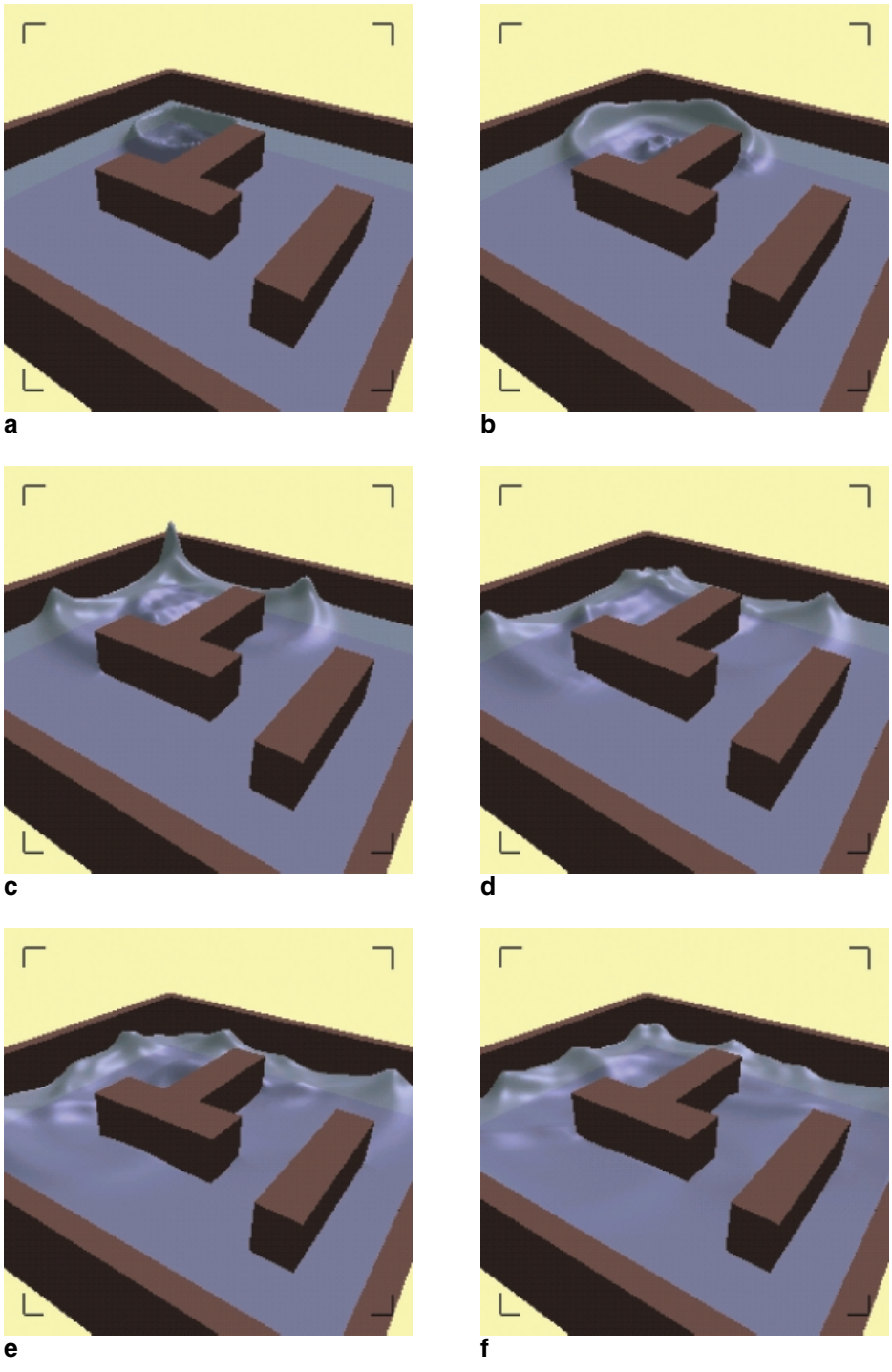

Fig. 4a–f. An example of wave animation with non-trivial boundary conditions. **a** First frame; **b** second frame; **c** third frame; **d** fourth frame; **e** fifth frame; **f** sixth frame

4.2 Solution stability and accuracy

Explicit time integration methods, because of their simplicity, were used in many implementations of fluid models (Chen and Lobo 1995; Chen et al. 1997; Foster and Metaxas 1996). However, the major disadvantage of explicit integration methods is that a 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 and third terms in (13) and (14), respectively). By adopting the semi-Lagrangian approach, we also ensure that the timestep size is not limited by the Courant–Friedrichs–Lewy (CFL) condition (Durran 1998). Our algorithm is stable as long as the departure points are estimated with sufficient accuracy. In other words, as long as the true departure points $(x_i - \alpha_i^n)$ fall between the two grid points (x_{m-1} and x_m) used in the linear interpolation (32) to estimate upstream function values, 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 (Pudykiewicz et al. 1985).

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 wavefront, and in 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 a hundred times larger than that for the explicit method. With very large timesteps, however, we observe noticeable damping, and the number of iterative steps required in the conjugate gradient solver becomes excessive as the initial guess becomes less accurate. Damping is caused by the implicit nature of the time integration scheme and by the spatial interpolation required to estimate the upstream function values. Nevertheless, 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 (without the Coriolis terms) 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 (1990), who based their model on a linearized version of the shallow water equations, also without the Coriolis terms. 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 timesteps.

4.3 Volume and energy conservation

We studied the conservative properties of our algorithm using the test cases mentioned in Sect. 4.2. Without any addition or removal of water, the total volume, computed by integrating water depth over the water domain, should remain constant throughout the simulation. In all our experiments, the change in water volume was found to be less than 5%. Since this variation is small and barely noticeable, we consider it acceptable.

On the other hand, our algorithm does not appear to conserve energy. The total energy E of the water is computed as the square of the magnitude of its velocity: $E = u^2 + v^2$. For a sufficiently long simulation, the waves eventually dampen out and the water surface returns to its calm and peaceful state. At first sight, this seems to be a disappointment, since there is no damping term in the shallow water equations, which implies that, once initiated, a wave should never die out. However, one must also take into account the effects of spatial interpolation and the implicit nature of the integration scheme, both of which introduce numerical damping, which causes the wave to lose its energy, slow down and eventually dampen out. This effect actually agrees (though for different reasons) with our everyday observation that ripples trapped in a finite pool dampen and slowly disappear. Moreover, real liquids lose energy because of their non-zero viscosity. The energy loss for our simulations occurs for the same reasons that contribute to the stability of the system: that velocity values are obtained by interpolating within the grid neighbourhood and that the integration is implicit. Thus, energy loss is a necessary compromise for achieving such a stable integration method.

4.4 Efficiency

In this section we give a simple analysis of the numerical complexity of our algorithm. Most of the effort of computation goes into solving the Helmholtz equation (25). Let N be the number of subintervals in one dimension, and k be the number of iterations of the conjugate gradient method, employed in solving (25). By taking advantage of the sparsity of the matrices, the conjugate gradient method has a complexity of $\mathcal{O}(N^2k)$ (Saad and Schultz 1986). 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 (1990). In their work, it was assumed that water speed varied slowly in space, allowing for a linearized version of the shallow water equations (8)–(10) to be used, one without the nonlinear advection terms. The 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. The form of ADI method presented by Kass and Miller (1990) appears to be unconventional in that the dependence of h on x and y is largely uncoupled, as represented by Kass and Miller's equations (19) and (20). A conventional (and improved) version of the ADI algorithm can be described as follows:
For $k = 1, 2, \ldots$, until convergence, do:

1. Solve the following equation in the x-direction:

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

2. Using newly computed $h^{n+\frac{1}{2},[k]}$, solve the following equation in the 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 Kass and Miller (1990). The linear systems that are solved in steps 1 and 2 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 Sect. 4.2, our algorithm was approximately four times as fast as that of Kass and Miller. This speed-up may be attributed to the faster convergence rate of the conjugate gradient method used in our algorithm over the ADI method chosen by Kass and Miller, and to the fact 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 (Gupta 1995). The semi-Lagrangian scheme introduces certain complications to parallel implementation on distributed memory machines, mainly that the departure points and the associated downstream points may not lie within the local memory of the same processor. Nevertheless, there have been a number of successful parallel implementations of semi-Lagrangian numerical methods (Malevsky and Thomas 1997; Thomas and Côté 1995; Thomas et al. 1997).

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 stable even with large timesteps. Compared to the model presented by Kass and Miller (1990), our algorithm uses fewer simplifying assumptions, since it is based on the complete shallow water equations (without the Coriolis terms), and is also more efficient, exhibiting a four-time speed-up in our experiments. With this speed-up, more detailed ocean scenes may be simulated in real time.
The limitations of our model lie in the fact 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 modelled. Nonetheless, our model serves well in its purpose of modelling 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 real-time 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 (Chen and Lobo 1995; Chen et al. 1997; Foster and Metaxas 1996). 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.

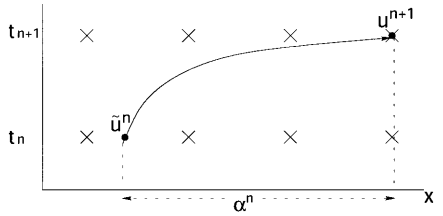

Fig. 5. An illustration of the semi-Lagrangian time integration method

Appendices

A Trajectory calculations

We describe in detail how the departure points are computed in Sect. 3.2. In (17) and (18), $\tilde{u}^n$ and $\tilde{h}^n$ are evaluated at *departure points* $(x_i - \alpha_i^n)$ at time level t_n in Fig. 5, where α_i^n is the displacement of a fluid particle in the time interval from t_n to t_{n+1}, ending at the downstream point x_i. The displacement α_i^n can be computed by integrating (15) backwards in time. With a first-order approximation,

$$\alpha_i^n = \Delta t u^n(x_i). \tag{31}$$

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 mth grid point: $x_m \equiv m\Delta x$. Suppose $x_{m-1} < x_i - \alpha_i^n < x_m$; then

$$\begin{aligned}\tilde{u}^n(x_i) &\equiv u^n(x_i - \alpha_i^n) \\ &= \frac{(x_i - \alpha_i^n - x_{m-1})u^n(x_m) + (x_m - x_i + \alpha_i^n)u^n(x_{m-1})}{\Delta x}.\end{aligned} \tag{32}$$

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

B Details of ADI iterations

We will now derive the linear systems that are solved in steps 1 and 2 of the ADI iterations outlined in Sect. 4.4. In the formulation of Kass and Miller (1990), 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 (see Fig. 1 in Kass and Miller (1990) for an illustration). After spatial discretization using a first-order finite difference method, the shallow water equations (8)–(10) become

$$\frac{\partial u_{i,j}}{\partial t} = -g\left(\frac{h_{i+1,j} - h_{i,j}}{\Delta x}\right), \tag{33}$$

$$\frac{\partial v_{i,j}}{\partial t} = -g\left(\frac{h_{i,j+1} - h_{i,j}}{\Delta y}\right), \tag{34}$$

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

Assuming that the water depth d is constant over each timestep, we take the time derivative of (35), eliminate from the resulting equation the dependence on u and v using (33) and (34), and obtain the following differential equation in h:

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

The goal of the first ADI subiteration (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:

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

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

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

Combining (36)–(39), we obtain the equation for the first ADI subiteration:

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

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

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

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

References

1. Chen JX, Lobo N da V (1995) Toward interactive-rate simulation of fluids with moving obstacles using Navier–Stokes equations. Graph Model Im Proc 57(2):107–116
2. Chen JX, Lobo N da V, Hughes CE, Moshell JM (1997) Real-time fluid simulation in a dynamic virtual environment. IEEE Comput Graph 17(3):52–61
3. Courant R, Isaacson E, Rees M (1952) On the solution of nonlinear hyperbolic differential equations by finite differences. Commun Pur Appl Math 5:243–255
4. Durran DR (1998) Numerical method for wave equations for geophysical fluid mechanics. Springer, Berlin Heidelberg New York
5. Foster N, Metaxas D (1996) Realistic animation of liquids. Graph Model Im Proc 58(5):471–483
6. Fournier A, Reeves WT (1986) A simple model of ocean waves. Comput Graph 20(4):75–84
7. Gupta A (1995) Performance and scalability of preconditioned conjugate gradient methods on parallel computers. IEEE Trans Parallel Distr Syst 6(5):455–469
8. Hestenes MR, Stiefel E (1952) Methods of conjugate gradient for solving linear systems. J Res Natl Inst Stan 45:409–436
9. Kang M, Fedkiw RP, Liu X-D (2000) A boundary condition capturing method for multiphase incompressible flow. UCLA Comput Appl Math Rep 99–27
10. Kass M, Miller G (1990) Rapid, stable fluid dynamics for computer graphics. Comput Graph 24(4):49–55
11. Khan R (1994) A simple model of ship wakes. MSc thesis, Dept Comput Sci, Univ Brit Columbia
12. Longuet-Higgin MS, Cokelet ED (1975) The deformation of steep surface waves on water I. A numerical method of computation. Proc R Soc Lond A 350:1–26
13. Longuet-Higgin MS, Cokelet ED (1978) The deformation of steep surface waves on water II. Growth of normal-mode instabilities. Proc R Soc Lond A 364:1–28
14. Malevsky AV, Thomas SJ (1997) Parallel algorithms for semi-Lagrangian advection. Int J Numer Meth Fluids 25:455–473
15. Peachey DR (1986) Modeling waves and surf. Comput Graph 20(4):65–74
16. Pudykiewicz J, Benoit R, and Staniforth A (1985) Preliminary results from a partial LRTAP model used on an existing meteorological forecast model. Atmos-Ocean 32:267–303
17. Robert A (1981) A semi-Lagrangian, semi-implicit numerical integration scheme for the primitive meteorological equations. Atmos-Oceans 19:35–46.
18. Saad Y, Schultz MH (1986) GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comp 7:856–869
19. Stam J (1999) Stable fluids. Comput Graph (SIGGRAPH '99 Proceedings) 121–128.
20. Staniforth A, Côté J (1991) Semi-Lagrangian integration schemes for atmospheric models – a review. Mon Wea Rev 119(9):2206–2223
21. Thomas and Côté(1995)]thomas95 Thomas SJ, Côté J (1995) Massively parallel semi-Lagrangian advection. J Simulat Pract Theory 3:223–238
22. Thomas SJ, Malevsky AV, Desgagné M, Benoit R, Pellerin P, Valin M. (1997) Massively parallel implementation of the mesoscale compressible community model. Parallel Comput 23:2143–2160

Photographs of the authors and their biographies are given on the next page.

ANITA T. LAYTON received her BSc degree in 1994 (Duke University), her MSc in 1996 (University of Toronto) and her PhD in 2001 (University of Toronto). She is currently a post-doctoral fellow at the Department of Mathematics of the University of North Carolina at Chapel Hill. Her research interests are in the computer animation of fluids, the development of efficient algorithms for solving fluid dynamics equations, including the shallow water equations and Navier–Stokes equations, the analysis of ocean models suitable for the study of climate, as well as mathematical renal modelling.

MICHIEL VAN DE PANNE is an associate professor in the Department of Computer Science at the University of Toronto. His research interests are in computer graphics, physics-based animation, and robotics. He has served on the program committees of ACM SIGGRAPH, ACM I3D, and Graphics Interface and co-chaired the 1997 Eurographics Workshop on Animation and Simulation. He is currently applying physics-based animation to the next generation of computer games at Motion Playground Inc. He obtained his BASc in 1987 (University of Calgary), his MASc in 1989 (University of Toronto), and his PhD in 1994 (University of Toronto).