Cheap Trickles

Jakob Egger[*]
IST Austria

Chris Wojtan[†]
IST Austria

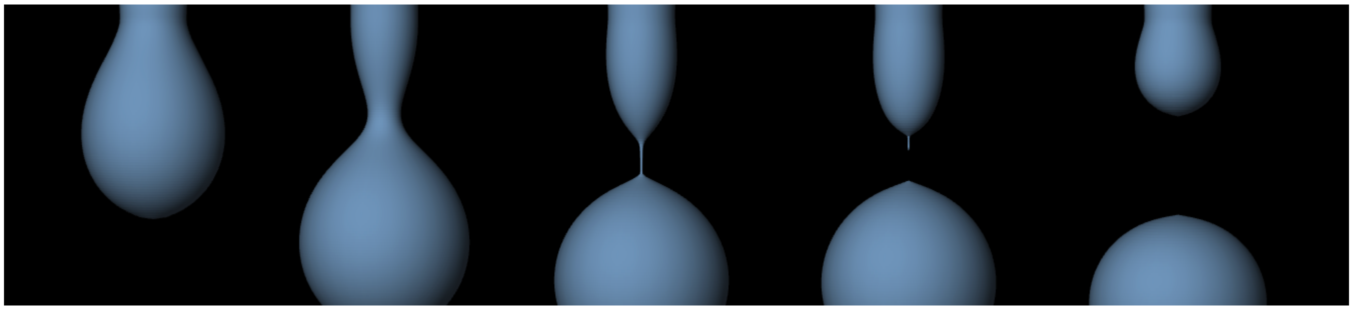

Figure 1: *By exploiting symmetries of the physical system, we can effectively reduce it to lower dimensions. This allows us to perform highly efficient yet detailed simulations of phenomena such as droplet pinch-off.*

Abstract

We suggest a novel approach to large scale fluid dynamic simulations with free surfaces. Rather than optimizing a generic method for arbitrary systems, we search for methods that are optimized for specific contexts such as thin sheets, spindles, or droplets. Our aim is to combine these distinct methods in a single simulation. As a first step, we present a method for handling thin spindles of fluid and their breakup into droplets.

Keywords: fluid jets, free surface flow, droplet pinch-off

1 Introduction

The major problem for free surface fluid simulations is the sheer amount of details required for convincing graphics. A single droplet falling onto a liquid surface can cause a crown splash with thin sheets dissolving into dozens of tiny droplets. With current methods, the number and size of these features is limited by the grid spacing. Even if we use an adaptive grid, it would be prohibitively expensive to simulate the hundreds and thousands of sheets and spindles and droplets caused by a breaking wave. But without these small features, the breaking wave will hardly be convincing; it will resemble a thick oily fluid.

On the other hand, some of these features on their own might be trivial to simulate. For example, a particle solver could easily simulate thousands of spherical droplets in free fall. In a similar manner, we could imagine simulating a spindle of liquid using a specialized method. We would not model it as three dimensional body of fluid, but rather as a curve in 3D space, a thread with a radius varying along its length. This would effectively reduce it to a single dimension and greatly reduce the computational complexity. Continuing down this path, a sheet of liquid could be modelled using a two-dimensional height field. Figure 2 illustrates this idea.

Inherent to all these models is that the level of detail can be increased to simulate effects at small scales without requiring a cubic increase in grid points. But it is even more exciting that this approach would allow to taylor each separate method to the application. For example, when simulating hundreds of tiny droplets,

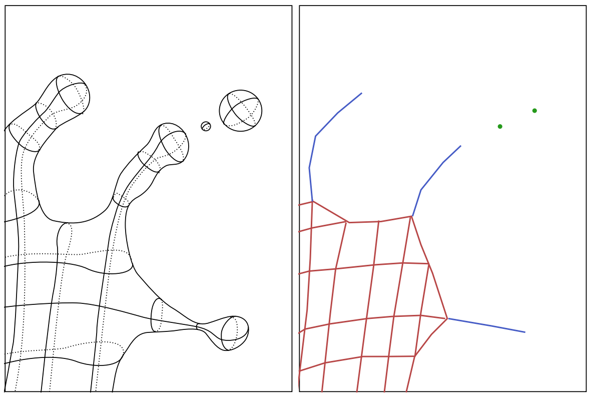

Figure 2: *Our ultimate goal is to create a fluid simulator that can automatically use different solvers based on the topology of the fluid. In the example above, the splash on the left could be modelled as a 2D sheet with connected 1D threads and disconnected droplets modelled as particles. The right half illustrates what the 'backbone' of this model would look like.*

oscillations of their surface will not be visible in the final graphic, but volume loss would look strange. So we would optimise the method to conserve volume.

As a first step towards this vision, we will look at the 1D case, a fluid jet. The behavior of fluid jets has been the subject of many previous works. Most notable for our application is an Eulerian scheme for threads with a vertical centerline developed by [Eggers and Dupont 1994]. Their model has been extended by [Lee et al. 2006] to allow for arbitrary centerlines in 3D. However, the method used by Lee et al required resampling at every simulation step, making their approach unsuitable for low viscosity fluids. A purely Lagrangian approach for high viscosity liquid threads was presented by [Bergou et al. 2010].

We are most interested in the behavior of surface tension driven effects in fluids with low viscosity. We therefore build on the work of Eggers and Dupont, and try to adapt their work to the requirements of computer graphics.

*e-mail: jakob.egger@ist.ac.at
†e-mail: chris.wojtan@ist.ac.at

2 A Model for Fluid Jets

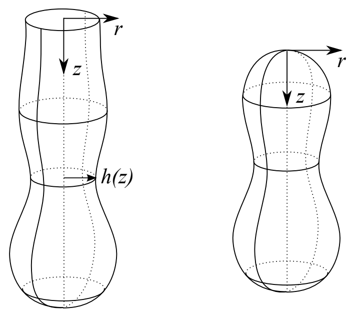

Figure 3: *Left: A jet with an 'inflow' boundary on top and free boundary on the bottom end; right: a droplet with free boundaries on both ends. The jets are symmetric around the z-axis, and the surface is defined by $h(z)$.*

We consider the geometry shown in figure 3. A jet of water flows along the vertical z-axis. [Eggers and Dupont 1994] introduced an approximation of the full Navier-Stokes equations that allows us to describe the jet using only one-dimensional functions: The axial velocity $v(z)$, the radius of the jet $h(z)$ and the pressure inside the jet $p(z)$. They derive the following set of differential equations:

$$\dot{v} = -vv' - \frac{1}{\rho}p' + g + 3\mu\left[v'' + 2\frac{h'}{h}v'\right] \tag{1}$$

$$p = \sigma\left[\frac{1 + h'^2 - hh''}{h\,(1 + h'^2)^{3/2}}\right] \tag{2}$$

$$\dot{h} = -vh' - \frac{1}{2}v'h \tag{3}$$

ρ denotes the density of the fluid, g is the gravitational acceleration, μ is the dynamic viscosity of the fluid, and σ is the surface tension.

We have explicit equations for $\dot{v}$ and $\dot{h}$. $\dot{v}$ consists of an advection term, a pressure term that essentially models surface tension forces, an external force, and a viscosity term. $\dot{h}$ consists of an advection term and a volume conservation term. The straightforward approach would be to simply use an explicit forward Euler integrator. However, the advection terms make explicit integration unstable. We found that applying a semi-Lagrangian advection scheme resolves the instability, and we were able to use explicit integration for the remaining terms.

2.1 Discretisation

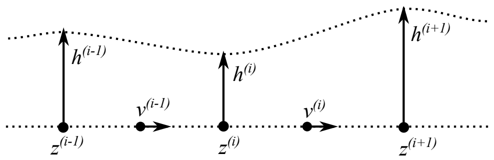

Figure 4: *Velocities are stored halfway between grid points.*

We chose a set of grid points $z^{(i)}$ that are not necessarily uniform. (Arbitrary grid points will make dealing with boundary conditions easier.) The radii $h^{(i)}$ are stored on the grid points. Since pressure p depends on h, h' and h'', we also evaluate the pressure values $p^{(i)}$ on the grid points. Beside advection, the most important term for v is the pressure term containing p'. We therefore decided to store the velocities $v^{(i)}$ at midpoints $\frac{1}{2}[z^{(i)} + z^{(i+1)}]$. This is also ideal when looking at the equation for h, as the volume conservation term depends on v'.

2.2 Advection

As stated above, h and v contain a nonlinear advection term that is prone to making the simulation unstable. A proven technique to handle this instability is to use semi-Lagrangian advection, or the 'method of characteristics' as described in [Stam 1999] and [Bridson 2008]. This method is very intuitive: To determine the new value of h or v at a specific grid point z, we simply determine where a particle at this location would have been at the previous time step, denoted by z_0. Then we take the value of h or v from this location. For backtracing the particle we use a second order Runge Kutta method:

$$z_{\text{mid}} = z - \frac{\delta t}{2}v(z) \tag{4}$$

$$z_0 = z - \delta t\, v(z_{\text{mid}}) \tag{5}$$

The problem with this method is that the z_0 is unlikely to be on a grid point, so we must interpolate the velocity and radius fields. This leads to numerical viscosity, and simple linear interpolation additionally causes severe volume loss. This can be alleviated by using Catmull-Rom interpolation (fitting a cubic spline to the values and derivatives). For stability, the interpolated values must be clipped to the neighboring values to prevent overshooting.

2.3 Pressure

The pressure p depends on h, h' and h''. The straightforward way is to determine h' and h'' using centered differences, and then calculate the pressure from that:

$$h'^{(i)} = \frac{h^{(i)} - h^{(i-1)}}{z^{(i)} - z^{(i-1)}} + \frac{h^{(i+1)} - h^{(i)}}{z^{(i+1)} - z^{(i)}} - \frac{h^{(i+1)} - h^{(i-1)}}{z^{(i+1)} - z^{(i-1)}}$$

$$h''^{(i)} = \frac{2}{z^{(i+1)} - z^{(i-1)}}\left[\frac{h^{(i+1)} - h^{(i)}}{z^{(i+1)} - z^{(i)}} - \frac{h^{(i)} - h^{(i-1)}}{z^{(i)} - z^{(i-1)}}\right]$$

$$p^{(i)} = \sigma\left[\frac{1 + {h'^{(i)}}^2 - h^{(i)}h''^{(i)}}{h^{(i)}\left(1 + {h'^{(i)}}^2\right)^{3/2}}\right] \tag{6}$$

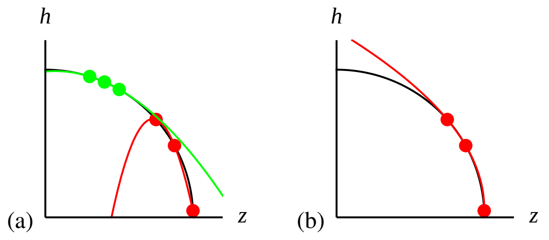

Figure 5: *(a) Vertical parabolas are fitted to three points on a circle. This is a good approximation in the middle (green), but not at the end (red). (b) Fitting a horizontal parabola instead works.*

However, this method breaks down when the radius become very steep, for example close to the end of a spherical drop. Centered difference basically fits a parabola of the form $h = k(z-z_0)^2 + d$ to three grid points, but in this case it would be better to fit a parabola of the form $z = k(h - h_0)^2 + d$. This is equivalent to using the centered difference formulas on the inverse of h to determine z' and

z''. Then we can use the well known formulas for the derivatives of inverse functions to calculate h' and h''.

$$z'^{(i)} = \frac{z^{(i)} - z^{(i-1)}}{h^{(i)} - h^{(i-1)}} + \frac{z^{(i+1)} - z^{(i)}}{h^{(i+1)} - h^{(i)}} - \frac{z^{(i+1)} - z^{(i-1)}}{h^{(i+1)} - h^{(i-1)}}$$

$$z''^{(i)} = \frac{2}{h^{(i+1)} - h^{(i-1)}} \left[\frac{z^{(i+1)} - z^{(i)}}{h^{(i+1)} - h^{(i)}} - \frac{z^{(i)} - z^{(i-1)}}{h^{(i)} - h^{(i-1)}} \right]$$

$$h'^{(i)} = \frac{1}{z'^{(i)}} \tag{7}$$

$$h''^{(i)} = \frac{z''^{(i)}}{(z'^{(i)})^3} \tag{8}$$

By choosing one of these two methods, we can reliably determine the pressure at all grid points.

2.4 Boundary conditions

We examine two kinds of boundary conditions. The first are fixed velocity and radius at the boundary (eg. a jet emerging from a nozzle). These are simple to implement: One simply does not change the radius and the adjacent velocity at the boundary.

The more complicated case is the free boundary (the end of a droplet). We must somehow keep track of where the surface of the droplet is. We do this by making the final grid point movable: it moves with the fluid. As soon as the distance of this boundary point to the next grid point becomes larger (or smaller) than a specific threshold, we insert (or remove) a grid point. The radius h at the boundary point is zero. Due to the staggered velocities, we don't need a velocity at the grid point, but we need a pressure value. We cannot use (6) because it diverges for $h = 0$. The simplest way is to fit a paraboloid to the end of the jet, with the apex at the boundary point. Then we can calculate the mean curvature at the tip and use that as the pressure at the endpoint:

$$p^{(0)} = 4\,\sigma \frac{z^{(1)} - z^{(0)}}{h^{(1)^2}} \tag{9}$$

This is all we need to update the velocity. But we still need a way to advect the free end point. Simply moving the end point with the adjacent velocity value works as long as the end point is receding, ie. moving inwards. As soon as the end point is moving away from the fluid, this method leads to significant volume loss. We compensated this effect with a numerical volume conservation term.

3 Results and Conclusion

We have written a Mac OS X application that implements the algorithm outlined in the previous section and provides an interactive interface for exploring the workings of the algorithm. Our test case consisted of a jet with an 'inflow' boundary condition on top, with a defined radius and inflow velocity. The fluid is accelerated downwards by gravity. Depending on inflow velocity, surface tension and viscosity, we might see droplets forming and pinching off, or a stable stream would evolve that only breaks up once it becomes thinner than some threshold radius.

Viscosity was the limiting factor for stability. Due to the explicit integration scheme, we needed to use very small time steps for large viscosities. However, there was also a limit on the other side: If we set viscosity to zero, the surface tension/pressure term blows up the simulation.

The most difficult part of the problem is the handling of free boundary conditions. While our method provides excellent results in the stationary case, there is still room for improvement when the boundary conditions move quickly. This is especially visible when looking at a droplet that is accelerated by gravity. The translation should ideally keep the profile of the droplet stationary, but numerical problems can cause substantial volume loss. We were able to compensate for this using a volume conservation term, but this term in turn can cause strange artefacts such as spikes protruding from otherwise perfectly spherical droplets.

In conclusion, we think that this method has great potential and demonstrates that it is possible to significantly reduce the complexity of fluid dynamics simulations when we take topology of the fluid into account. This specialized model for jets retains many important effects such as droplet pinch-off at a fraction of the complexity of a full solver. If we can extend this model to arbitrary centerlines in 3D, we will be able to use it to supplement a full solver for efficient handling of small details.

References

BERGOU, M., AUDOLY, B., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2010. Discrete viscous threads. *ACM Transactions on Graphics (TOG) 29*, 4, 116.

BRIDSON, R. 2008. *Fluid simulation for computer graphics*. Ak Peters Series. A K Peters.

EGGERS, J., AND DUPONT, T. 1994. Drop formation in a one-dimensional approximation of the navier–stokes equation. *Journal of fluid mechanics 262*, 1, 205–221.

LEE, S., OLSEN, S., AND GOOCH, B. 2006. Interactive 3d fluid jet painting. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, ACM, 97–104.

STAM, J. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, 121–128.