Project CFD–Lab
Free Surface Simulation

Group Number: 1
Ioannis Kouroudis, Lukas Krenz, Andreas Reiser,

Supervisor
Friedrich Menhorn

August 1, 2017

Abstract

In industry there are many applications that require the simulation of a system of unmixable fluids with vastly different kinematic viscosity, separated by a discrete interface. This problem is commonly referred to as a free surface flow problem. In effect, the kinematic viscosity difference causes the first fluid, which is usually the industrially relevant one, to exert force on the second one, receiving a negligible reaction in return. The most common configuration of this sort is a liquid-gas system and is frequently encountered in applications such as dam breaking scenarios, material processes and wave simulations. In this particular project, the phenomenon is simulated using a Lattice Boltzmann approach combined with a volume of fluid method.

1 Aspects of implementation

The algorithm generally works by keeping track of the mass exchange between cells. To do this, we save the mass m for each cell and convert the cells if they are either full or empty. Between the filled and empty cells we insert interface cells. The program was written in C++ utilising OpenMP parallelization. We followed the algorithm described in [Thü07].

1.1 Gravity

We modify the standard collision process to include the gravity force $\boldsymbol{g}$ [TPR+06]

$$f_i(x, t + \Delta t) = (1 - \omega) f_i^{eq} + \omega f_i^{eq}(\rho(\mathbf{x}, t), \mathbf{u}(\mathbf{x}, t) + \frac{g}{\omega}). \tag{1}$$

1.2 Mass exchange

In order to keep track of the interface, two additional variables have to be stored for each cell, namely the mass m and the fluid fraction $\varepsilon_i = \frac{m_i}{\rho_i}$. The

mass exchanged between an interface and a fluid cell can be calculated directly using their distributions

$$\Delta m_i(x, t + \Delta t) = f_{\hat{i}}(x + e_i, t) - f_i(x, t), \tag{2}$$

where $\hat{i}$ is the inverse velocity of i. For interface–interface mass exchange we must take the fluid fraction of both cells into account.

$$\Delta m_i(x, t + \Delta t) = s_e \frac{\varepsilon(x + \Delta t e_i, t) + \varepsilon(x, t)}{2} \tag{3}$$

To avoid "lonely" interface cells that have no empty or filled neighbours, we use the appropriate s_e from table 1 in (3). Because the mass exchange is a symmetric process, the used scheme is explicitly mass preserving.

	standard cell at $(x + e_i)$	no fluid nb at $(x + e_i)$	no empty nb at $(x + e_i)$
standard cell at (x)	$f_{\hat{i}}(x_{nb}, t) - f(x, t)$	$f_{\hat{i}}(x_{nb}, t)$	$-f_i(x, t)$
no fluid neighbours at (x)	$-f_i(x, t)$	$f_{\hat{i}}(x_{nb}, t) - f(x, t)$	$-f_i(x, t)$
no empty neighbours at (x)	$f_{\hat{i}}(x_{nb}, t)$	$f_{\hat{i}}(x_{nb}, t)$	$f_{\hat{i}}(x_{nb}, t) - f(x, t)$

Table 1: Interface–Interface mass streaming coefficient s_e with x_b as the neighbouring cell.

1.3 Boundary conditions

Because interface cells have empty cells as neighbours, they do not have a full set of distributions. This is why we need to reconstruct all distributions coming from empty cells during the streaming step. The physical perspective of this condition is that the gas has a far greater kinematic viscosity and is consequently pushed without resistance by the liquid. Further, it is assumed that the gas pressure is atmospheric, which is used in place of density in the equilibrium function($\rho_A = 1$). If e_i comes from an empty cell into a interface cell the following distribution is constructed:

$$f'_{\hat{i}}(x, t + \Delta t) = f^{eq}_{\hat{i}}(\rho_A, \mathbf{u}) + f^{eq}_i(\rho_A, \mathbf{u}) - f_i(\mathbf{x}, t). \tag{4}$$

However, constructing these distributions along only one side of the interface would result in a force imbalance. To rectify this, we additionally reconstruct distributions coming in along the surface normal, which we approximate by central differences

$$\mathbf{n} = \frac{1}{2} \begin{bmatrix} \varepsilon(x_{j-1,k,l}) - \varepsilon(x_{j+1,k,l}) \\ \varepsilon(x_{j,k-1,l}) - \varepsilon(x_{j,k+1,l}) \\ \varepsilon(x_{j,k,l-1}) - \varepsilon(x_{j,k,l+1}) \end{bmatrix}, \tag{5}$$

where $\varepsilon(x_{j,k,l})$ is the fluid fraction of the cell at coordinates (j, k, l). If $(n \cdot e_i) > 0$, f_i is reconstructed according to equation (4)

1.4 Flag Update

After the mass calculation step, all the interface cells that have filled or emptied are assigned temporary flag as filled or emptied cells respectively. To determine this, the criterion

$$m(x, t + \Delta t) > (1 + \text{offset }) \cdot \rho(x, t + \Delta t), \tag{6}$$

$$m(x, t + \Delta t) < (0 - \text{offset }) \cdot \rho(x, t + \Delta t), \tag{7}$$

is used, with an offset of 10^{-3}, to avoid oscillating changes of a cell status [Thü07].

We then convert all empty neighbours of filled cells to interface cells and all neighbouring emptied cells back to interface cells. After this is done, all fluid neighbours of emptied cells are converted to interface cells. Formerly empty cells that became interface cells are initialised with the average density and velocity of their fluid, interface and filled neighbours. We need to use temporary flags because otherwise the update rule would be order dependent.

1.5 Excess Mass

Newly converted cells have an excess mass m^{ex}. Fluid cells should have mass equal to their density, empty cells should have mass zero. All excess mass is distributed to all interface neighbours. For reasons of stability, we weigh the update according to the position of the current interface

$$\Delta m(x + e_i) = m^{\text{ex}} \frac{v_i}{v_{\text{total}}}, \tag{8}$$

where v_i is the weight of the neighbouring cell and v_{total} is the sum of all weights. The weighs are calculated by

$$v_i^f = \max(0, \boldsymbol{n} \cdot \boldsymbol{e_i}), \tag{9}$$

$$v_i^e = -\min(0, \boldsymbol{n} \cdot \boldsymbol{e_i}), \tag{10}$$

for filled and emptied cells respectively.

Again it is important to update the mass of all cells at the same time. After the excess mass has been distributed, all flags are converted from the temporary to the finalised ones.

1.6 Adaptive time step and turbulence model

To avoid instabilities due to high velocities and to acquire speedup during periods of rest we use an adaptive time step. We use a critical velocity of $u_{\text{crit}} = 1/6$, which is half the lattice speed of sound. If velocities become larger than $1/3$ distributions can become negative thus impacting the stability.

To do this, we calculate the maximum velocity $u_{\max}$ of the simulation and modify the time step if needed. If $\|u_{max}\| > \xi u_{\text{crit}}$ the new timestep becomes $\Delta t_{\text{new}} = \xi \Delta t_{\text{old}}$ and if $\|u_{max}\| < \xi u_{\text{crit}}$ the new timestep is updated to $\Delta t_{\text{new}} = \Delta t_{\text{old}}/\xi$, where $\xi = 4/5$ is the rescaling factor [TRK05]. The timestep is used to compute a new value of the relaxation parameter τ

$$\tau_{\text{new}} = \frac{\Delta t_{\text{new}}}{\Delta t_{\text{old}}}(\tau_{\text{old}} - 0.5) + 0.5 \tag{11}$$

which is then used in the collision process.

After each time step change, all time dependent values must be rescaled, e.g. density, mass, velocity and gravity. Additionally all distributions must be rescaled. The details of the rescaling procedure are omitted to preserve space, the interested reader is referred to chapter 6 of [Thü07].

For very small time steps or viscosities, τ is close to 0.5, thus leading to instabilities in the simulation. To rectify this, we use a Smagorinsky turbulence model. For this we calculate the local stress tensor S by

$$\Pi_{\alpha,\beta} = \sum_{i=1}^{19} e_{i\alpha} e_{i\beta} (f_i - f_i^{eq}), \tag{12}$$

where α and β are the components of the lattice velocities. The intensity S is then given by

$$S = \frac{1}{6C^2} (\sqrt{\nu^2 + 18C^2 \sqrt{\Pi_{\alpha,\beta}\Pi_{\alpha,\beta}}} - \nu, \tag{13}$$

where summation over α and β is implied and C is a constant usually set to values close to 0.03 [Thü07]. Finally we set the local relaxation time to

$$\tau_s = 3(\nu + C^2 S) + \frac{1}{2}, \tag{14}$$

where ν is the global viscosity. Intuitively, the turbulence model increases the local viscosity for cells according to the stress exerted onto them.

2 Results/Scenario

Two simple scenarios were chosen, and as can be seen, produced very good physical results.

2.1 Dam Breaking

The dam breaking was simulated by having a large vertical mass of fluid at the start be released for one side of the geometry.

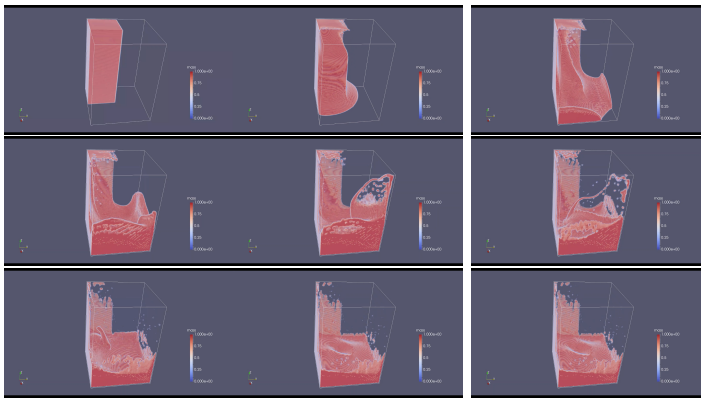

Figure 1: Breaking Dam

2.2 Falling Drop

A drop was left to fall in a pool of stationary water. As can be seen, the results look comparable to other sources and resemble the expected physical behaviour.

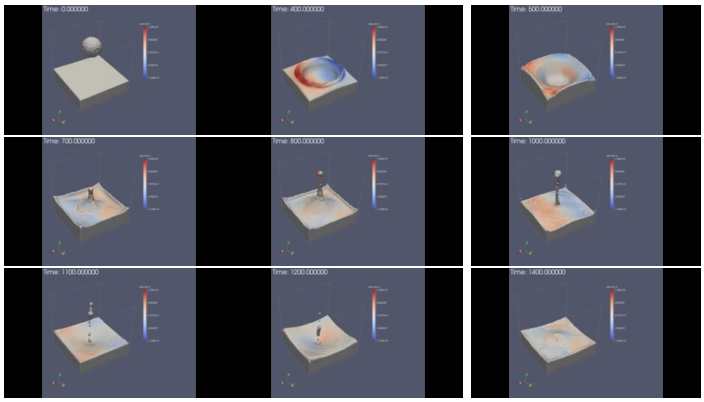

Figure 2: Falling Drop

References

[Thü07] Nils Thürey. *Physically Based Animation of Free Surface Flows with the Lattice Boltzmann Method: Physikalische Animation Von Strömungen Mit Freien Oberflächen Mit Der Lattice-Boltzmann-Methode*. Verlag Dr. Hut, 2007.

[TPR+06] Nils Thürey, Thomas Pohl, Ulrich Rüde, Markus Oechsner, and Carolin Körner. Optimization and stabilization of lbm free surface flow simulations using adaptive parameterization. *Computers & fluids*, 35(8):934–939, 2006.

[TRK05] Nils Thürey, Ulrich Rüde, and Carolin Körner. Interactive free surface fluids with the lattice boltzmann method. *Technical Report05-4. University of Erlangen-Nuremberg, Germany*, 2005.